**ERROR-CORRECTING CODES (ECC)**

Consider the following binary digital communication system:

![Diagram of binary digital communication system]

where $m_n$, $c_n$, $s_n$, $r_n$, $v_n = \hat{c}_n$, and $\hat{m}_n$ is the $n^{th}$ message bit, coded bit, transmitted signal, received noisy signal, demodulated coded bit, and decoded message bit, respectively.

The error-correcting code (ECC) channel encoder adds controlled redundancy to the message bit sequence, so that up to some limit, data errors that occur during transmission and reception can be corrected and the original information (message) sequence recovered.

In this course we will study convolutional error correcting codes only.
Convolutional Codes

Unlike linear block encoders, convolutional encoders operate continuously on a serial stream of message bits (sequentially). Also, to generate a code word n bits long from a k-bit long message sequence, the output of the encoder at any given time depends not only on the k-bit long information data, but also on a previous block of input bits, whose length depends on the memory depth of the encoder.

Consider the following encoder:
where \( x \triangleq \text{unit-delay} \)

To start the encoding process, reset the shift registers (set them to 0) of the encoder and let the bit stream \((m_0, m_1, m_2) = (110)\) be input sequentially, then

1. \( m_0 = 1 \Rightarrow c_0^1 = c_0^2 = 1 \Rightarrow \text{output} = 11 \)
2. \( m_1 = 1 \Rightarrow c_1^1 = 0, c_1^2 = 1 \Rightarrow \text{output} = 01 \)
3. \( m_2 = 0 \Rightarrow c_2^1 = 0, c_2^2 = 1 \Rightarrow \text{output} = 01 \)

This encoder outputs 2 coded bits for every message input bit \( \Rightarrow \) code rate \( r = \frac{1}{2} \).

Define the constraint length \( \nu \) of the code as \( \nu \triangleq 1 + \# \text{ of past inputs that affect the current outputs} \).

Let \((m_0, m_1, m_2, \ldots)\) denote the input message sequence to a single-input, n-output, one bit at a time. Let \( (g_0^j, g_1^j, \ldots, g_M^j) \) denote the unit-sample response of the \( j \)th input-output path, \( j = 1, 2, \ldots, n \), i.e., they are the responses to the input sequence \( (1 \ 0 \ 0 \ \ldots) \).
Let $\{c_i^j\}$ be the output sequence generated by the $j^{th}$ input-output path of the rate $r = \frac{1}{n}$ encoder. Then
\[
c_i^j = \sum_{k=0}^{M} g_k^j m_{i-k}, \quad i = 0, 1, 2, \ldots, j = 0, 1, 2, \ldots, n, \quad m_{i-k} = 0, \quad k > i,
\]
that is, $c_i^j$ is the modulo-2 convolution sum of the input sequence and the unit-sample response of the $j^{th}$ path. $M$ is the memory depth.

After the convolution, the sequences $\{c_i^1\}, \{c_i^2\}, \ldots, \{c_i^n\}$ are combined by a multiplexer to produce the output coded sequence $\left( c_0^1 c_0^2 \ldots c_0^n, c_1^1 c_1^2 \ldots c_1^n, \ldots \right)$.

**Example:** Consider the previous rate $\frac{1}{2}$ encoder. Then $M = 2$, $g^{(1)} = \left( g_0^1 g_1^1 g_2^1 \right) = (111)$ and $g^{(2)} = \left( g_0^2 g_1^2 g_2^2 \right) = (101)$. For the input sequence $\{m_i\} = (m_0 \, m_1 \, m_2) = (110)$,
\[
c_i^j = \sum_{k=0}^{2} g_k^j m_{i-k}, \quad j = 1, 2
\]
Explicitly,

\[ c_0^1 = \sum_{k=0}^{2} g_k^1 m_{0-k} = g_0^1 m_0 = 1 \cdot 1 = 1 \]

\[ c_1^1 = \sum_{k=0}^{2} g_k^1 m_{1-k} = g_0^1 m_1 + g_1^1 m_0 = 1 \cdot 1 + 1 \cdot 1 = 0 \]

\[ c_2^1 = \sum_{k=0}^{2} g_k^1 m_{2-k} = g_0^1 m_2 + g_1^1 m_1 + g_2^1 m_0 = 1 \cdot 0 + 1 \cdot 1 + 1 \cdot 1 = 0 \]

\[ c_3^1 = \sum_{k=0}^{2} g_k^1 m_{3-k} = g_1^1 m_2 + g_2^1 m_1 = 1 \cdot 0 + 1 \cdot 1 = 1 \]

\[ c_4^1 = \sum_{k=0}^{2} g_k^1 m_{4-k} = g_2^1 m_2 = 1 \cdot 0 = 0 \]

or \[ \{c_i^1\} = (1 0 0 1 0) \].

Likewise, \[ \{c_i^2\} = (1 1 1 1 0) \] and the multiplexed sequence is \((11, 01, 01, 11, 00)\).

To restore the zero initial state of the shift register, a tail of \(M\) zeros has to be appended to the input message sequence.

Consider now the two-input three-output encoder in the following diagram:
From the diagram, the unit sample responses from input 1 to all 3 outputs are described by

\[ g_1^{(1)} = (g_{1,0}^1 g_{1,1}^1) = (11), \quad g_1^{(2)} = (g_{1,0}^2 g_{1,1}^2) = (01), \quad \text{and} \quad g_1^{(3)} = (g_{1,0}^3 g_{1,1}^3) = (11). \]

and unit sample responses from input 2 to all 3 outputs are described by

\[ g_2^{(1)} = (g_{2,0}^1 g_{2,1}^1) = (01), \quad g_2^{(2)} = (g_{2,0}^2 g_{2,1}^2) = (10), \quad \text{and} \quad g_2^{(3)} = (g_{2,0}^3 g_{2,1}^3) = (10). \]
Therefore,
\[ c^{(1)} = m^{(1)} \ast g_1^{(1)} \oplus m^{(2)} \ast g_2^{(1)} \]
\[ c^{(2)} = m^{(1)} \ast g_1^{(2)} \oplus m^{(2)} \ast g_2^{(2)} \]
\[ c^{(3)} = m^{(1)} \ast g_1^{(3)} \oplus m^{(2)} \ast g_2^{(3)} \]
and \( \{c_i\} = (c_0^{(1)}, c_0^{(2)}, c_0^{(3)}, c_1^{(1)}, c_1^{(2)}, c_1^{(3)}, c_2^{(1)}, c_2^{(2)}, c_2^{(3)}, \ldots) \).

Let \( k \) be the number of information bits input to the encoder and let \( l_i \) be the length (number of memory elements) of the \( i \)th shift register. Then the encoder memory order \( m_d \) is defined by
\[ m_d = \max_{1 \leq i \leq k} \{l_i\} \]
and the constraint length by
\[ \nu = m_d + 1. \]

\( m_d \) and \( \nu \) in the last two encoders are 2 and 3, and 1 and 2, respectively.
Formally, the memory depth $m_d$ of a $\frac{1}{n}$ encoder is defined by

$$m_d \triangleq \max_j \{\deg[g^{(j)}(x)]\}, \ j = 1, \ldots, n$$

Since $k$ information bits generate $n$ convolutionally encoded bits, the ratio $R = k / n$ is called the code rate. Clearly, if the encoder input bit rate is $R_b$, then the output rate is $(n / k)R_b$. Ideally, we would like the code rate $R$ to be close to 1, in order to minimize transmission bandwidth requirements. In reality, a trade off has to be made between the necessary redundancy to achieve a certain BER and the minimization of bandwidth occupancy.

An alternative representation of the convolutional encoding operation is one that uses the transform domain (analogous to the frequency domain). This comes from the realization that convolution in one domain is equivalent to multiplication in the transform domain.
In the transform domain, each path of a rate $1/n$ convolutional encoder can be described in terms of an FIR filter, i.e., the $j^{th}$ path is described by

$$g^{(j)}(x) = g_0^j + g_1^j x + \ldots + g_M^j x^M, \ j = 1, 2, \ldots, n,$$

where $x$ is a unit-delay operator. The polynomials $g^{(j)}(x), \ j = 1, 2, \ldots, n$, are called the generator polynomials of the code and their coefficients $g_k^j \in GF(2)$ are such that,

$$g_k^j = \begin{cases} 1, & \text{if there is a connection to the summing junction} \\ 0, & \text{if there is no connection to the summing junction} \end{cases},$$

where $GF(2)$ is a binary Galois Field.

Moreover, in the transform domain, the message sequence can be described by

$$m(x) = m_0 + m_1 x + \cdots + m_{l-1} x^{l-1},$$

where $l$ is the length of the message. The input-output relation of the $j^{th}$ path is given by

$$c^{(j)}(x) = m(x) g^{(j)}(x), \ j = 1, 2, \ldots, n,$$

in modulo-2 arithmetic.
After multiplexing, for a rate $\frac{1}{2}$ encoder, $c(x) = c^{(1)}(x^2) \oplus xc^{(2)}(x^2)$.

Example: For the rate $r = \frac{1}{2}$, $m_d = 2$ encoder in question, $g^{(1)}(x) = 1 + x + x^2$ and $g^{(2)}(x) = 1 + x^2$. With the input sequence $(m_0 m_1 m_2) = (101)$ or $m(x) = 1 + x$,

$$c^{(1)}(x) = m(x) g^{(1)}(x) = (1 + x)(1 + x + x^2) = 1 + x + x^2 + x + x^2 + x^3 = 1 + x^3.$$  

Thus, $\{c_i^{(1)}\} = (1001)$.

$$c^{(2)}(x) = m(x) g^{(2)}(x) = (1 + x)(1 + x^2) = 1 + x^2 + x + x^3 = 1 + x + x^2 + x^3$$

Hence, $\{c_i^{(2)}\} = (1111)$.

In polynomial form (in the transform domain), the encoded sequence is given by

$$c(x) = c^{(1)}(x^2) \oplus xc^{(2)}(x^2) = 1 + x^6 + x(1 + x^2 + x^4 + x^6) = 1 + x + x^3 + x^5 + x^6 + x^7.$$  

Finally, in binary form, the encoded sequence is given by $\{c_i\} = (11010111)$.  

For a \((n, k)\) convolutional code, 
\[ \zeta(x) = m(x)G(x), \]
where

\[
\begin{align*}
m(x) &= \begin{bmatrix} m^{(1)}(x) & m^{(2)}(x) & \cdots & m^{(k)}(x) \end{bmatrix}, \\
G(x) &= \begin{bmatrix} g^{(1)}_1(x) & g^{(2)}_1(x) & \cdots & g^{(n)}_1(x) \\
g^{(1)}_2(x) & g^{(2)}_2(x) & \cdots & g^{(n)}_2(x) \\
\vdots & \vdots & \ddots & \vdots \\
g^{(1)}_k(x) & g^{(2)}_k(x) & \cdots & g^{(n)}_k(x) \end{bmatrix}, \\
\zeta(x) &= \begin{bmatrix} c^{(1)}(x) & c^{(2)}(x) & \cdots & c^{(n)}(x) \end{bmatrix}.
\end{align*}
\]

After multiplexing,
\[
c(x) = c^{(1)}(x^n) \oplus xc^{(2)}(x^n) \oplus \cdots \oplus x^{n-1}c^{(n)}(x^n).
\]

For the previous \((n,k) = (3, 2)\) convolutional code,
\[
G(x) = \begin{bmatrix} 1+x & x & 1+x \\
x & 1 & 1 \\
x & 1 & 1 \end{bmatrix}.
\]
If \( \{m_i^{(1)}\} = (101) \) or \( m^{(1)}(x) = 1 + x^2 \) and \( \{m_i^{(2)}\} = (110) \) or \( m^{(2)}(x) = 1 + x \), then

\[
\zeta(x) = \begin{bmatrix} c^{(1)}(x) & c^{(2)}(x) & c^{(3)}(x) \end{bmatrix} = m(x)G(x) = \begin{bmatrix} m^{(1)}(x) & m^{(2)}(x) \\
g^{(1)}_1(x) & g^{(2)}_1(x) & g^{(3)}_1(x) \\
g^{(1)}_2(x) & g^{(2)}_2(x) & g^{(3)}_2(x) \end{bmatrix}
\]

\[
= \begin{bmatrix} 1 + x^2 & 1 + x \\
1 + x & x \\
x & 1 \end{bmatrix} = \begin{bmatrix} 1 + x^3 & 1 + x^3 & x^2 + x^3 \end{bmatrix}
\]

and

\[
c(x) = c^{(1)}(x^3) \oplus xc^{(2)}(x^3) \oplus x^2c^{(3)}(x^3) = 1 + x^9 + x(1 + x^9) + x^2(x^6 + x^9) = 1 + x + x^8 + x^9 + x^{10} + x^{11}
\]

Hence, \( \{c_i\} = (110000001111) \).

Finally, one can show that for any \((n, k)\) encoder,

\[
c(x) = \sum_{i=1}^{k} m^{(i)}(x^n) g_i(x),
\]

where \( g_i(x) = g_i^{(1)}(x^n) + xg_i^{(2)}(x^n) + \cdots + x^{n-1}g_i^{(n)}(x^n), 1 \leq i \leq k \).
Example: For the $(n, k) = (2, 1)$ encoder of the first example,

\[
g_1(x) = g_1^{(1)}(x^2) + xg_1^{(2)}(x^2) = 1 + x^2 + x^4 + x(1 + x^4) = 1 + x + x^2 + x^4 + x^5.
\]

With $\{m_i\} = (10011)$, $m^{(1)}(x) = 1 + x^3 + x^4$ and $m^{(1)}(x^2) = 1 + x^6 + x^8$. Hence,

\[
c(x) = m^{(1)}(x^2)g_1(x) = (1 + x^6 + x^8)(1 + x + x^2 + x^4 + x^5)
= 1 + x + x^2 + x^4 + x^5 + x^6 + x^7 + x^9 + x^{11} + x^{12} + x^{13}
\]

and $\{c_i\} = (11101110101111)$.

The generator matrix of a rate $\frac{1}{n}$ convolutional code can be expressed by

\[
G(x) = \begin{bmatrix} g^{(1)}(x) & g^{(2)}(x) & \cdots & g^{(n)}(x) \end{bmatrix}.
\]
Therefore,
\[
c(x) = \begin{bmatrix} c^{(1)}(x) | c^{(2)}(x) | \cdots | c^{(n)}(x) \end{bmatrix} = \begin{bmatrix} m(x) g^{(1)}(x) | \cdots | m(x) g^{(n)}(x) \end{bmatrix} = m(x) \begin{bmatrix} g^{(1)}(x) | \cdots | g^{(n)}(x) \end{bmatrix} = m(x) G(x).
\]

**Definition:** A convolutional code is said to be **catastrophic** if a non-zero input sequence to the encoder results, after some finite time, in an all-zero output sequence.

**Definition:** A rate \( r = \frac{1}{n} \) convolutional code with code generator matrix \( G(x) \) is **not catastrophic** if and only if
\[
\text{GCD} \left\{ g^{(j)}(x), \ j = 1, 2, \ldots, n \right\} = x^\ell,
\]
for some integer \( \ell \geq 0 \).
Example: Consider the following rate $r = \frac{1}{2}$ encoder:

\[
\{m_i\} \quad x \quad x \quad x \quad \{c_i^1\} \\
\{c_i^2\}
\]

If $\{m_i\} = \{1, 1, 1, \ldots\}$, then $\{c_i\} = \{c_i^1, c_i^2\} = \{11, 00, 11, 00, 00, \ldots\}.$
Example: For the encoder of the previous example,

\[
G(x) = \left[1 + x + x^2 + x^3 \right] \left[1 + x + x^2 + x^3 \right] \implies G \subset D \left\{ g^{(j)}(x), j = 1,2 \right\} = (1+x)^3
\]

since \( 1 + x + x^2 + x^3 = (1+x)^3 \) implies that the code generated by the encoder is catastrophic.

**Systematic Convolutional Codes:**
They are a subclass of convolutional codes whose first \( k \) coded bits in the \( n \)-long output code word are exact replicas of the \( k \) input information bits. That is,

\[
c_i^{(1)} c_i^{(2)} \cdots c_i^{(k)} c_i^{(k+1)} \cdots c_i^{(n)} = m_i^{(1)} m_i^{(2)} \cdots m_i^{(k)} c_i^{(k+1)} \cdots c_i^{(n)}.
\]
The transfer function matrix for these codes is

\[
G(x) = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 & G_1^{(k+1)}(x) & \cdots & G_1^{(n)}(x) \\
0 & 1 & 0 & \cdots & 0 & G_2^{(k+1)}(x) & \cdots & G_2^{(n)}(x) \\
\vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 0 & G_{k-1}^{(k+1)}(x) & \cdots & G_{k-1}^{(n)}(x) \\
0 & 0 & 0 & \cdots & 1 & G_k^{(k+1)}(x) & \cdots & G_k^{(n)}(x)
\end{bmatrix} = \begin{bmatrix} I_k & G_k(x) \end{bmatrix},
\]

where \( I_k \) is a \( k \times k \) identity matrix and \( G_k(x) \) is a \( k \times (n-k) \) parity check matrix. Note that the last \( n-k \) coded bits are called the parity bit sequence.
Example: Consider the following systematic convolutional encoder:

\[
G(x) = \begin{bmatrix}
1 & 1 + x + x^3
\end{bmatrix}.
\]

If \( \{m_i\} = (1011) \) or \( m(x) = 1 + x^2 + x^3 \), then the output of the encoder is given by

\[
\zeta(x) = m(x)G(x) = \left(1 + x^2 + x^3\right)\begin{bmatrix}
1 & 1 + x + x^3
\end{bmatrix} = \begin{bmatrix}
1 + x^2 + x^3 & 1 + x + x^2 + x^3 + x^4 + x^5 + x^6
\end{bmatrix}.
\]

In polynomial form in the transform domain, the output is described by

\[
c(x) = c^{(1)}(x^2) \oplus xc^{(2)}(x^2) = 1 + x + x^3 + x^4 + x^5 + x^6 + x^7 + x^9 + x^{11} + x^{13}.
\]
Thus, the output sequence is given by \( \{c_i\} = \{1,1,0,1,1,1,0,1,0,1\} \), where the bits with the underbar are the message bits.

An advantage of systematic convolutional codes is that the encoders are simpler to implement than the nonsystematic counterparts. Also, systematic codes do not require an inverting circuit to recover the information bit sequence from the code word. That is, for nonsystematic codes, an \( n \times k \) matrix \( G^{-1}(x) \) must exist so that

\[
c(x)G^{-1}(x) = m(x)G(x)G^{-1}(x) = m(x)x',
\]

for some \( l \geq 0 \).

A consequence of this fact is that a feed forward inverse \( G^{-1}(x) \) exists (with delay \( l \)) if and only if

\[
GCD \left\{ \Delta_i(x) : i = 1, 2, \ldots, \binom{n}{k} \right\} = x', \quad l \geq 0,
\]

where \( \Delta_i(x) \) are the determinants of the \( \binom{n}{k} \) distinct \( k \times k \) submatrices of the transfer function matrix \( G(x) \).
Example: The generator matrix of the \((2, 1)\) code of the first example is
\[
G(x) = \begin{bmatrix} 1 + x + x^2 & 1 + x^2 \end{bmatrix}.
\]

Thus, \(GCD\{G(x)\}\) is \(x^0 = 1\), which means that \(G^{-1}(x)\) exists. In fact,
\[
G^{-1}(x) = \begin{bmatrix} 1 + x + x^2 \\ x^2 \end{bmatrix}.
\]

A severe consequence of the nonexistence of the feed forward inverse \(G^{-1}(x)\) is that the code generated by the encoder with transfer function matrix \(G(x)\) can have the undesirable characteristic of mapping an infinite weight information bit sequence into a finite weight code word.

Example: Consider the convolutional encoder
By inspection, \( G(x) = \left[ x + x^2 + x^3 \quad 1 + x + x^2 \right] \).

\[
GCD \{G(x)\} = 1 + x + x^2 \neq x', \quad \forall l \geq 0.
\]

Let \( m(x) = \frac{1}{1 + x + x^2} = 1 + x + x^3 + x^4 + x^6 + x^7 + x^9 + x^{10} + \cdots \). Then \( \zeta(x) = [x \quad 1] \) and 
\[
c(x) = c^{(1)}(x^2) \oplus xc^{(2)}(x^2) = x + x^2.
\]
This means that the information sequence described by \( \{m_i\} = (11011011011\cdots) \) gets mapped onto the sequence \( \{c_i\} = (011000\cdots) \), a sequence of Hamming weight (the number of 1’s in the sequence) 2. The problem here is that two channel errors can produce an infinite (or large) number of errors in the decoder. **Definition:** A convolutional code is said to be catastrophic if there exists an input sequence \( \{m_i\} \) such that its Hamming weight is infinite, i.e. \( H_w (\{m_i\}) = \infty \) and the Hamming weight of the output sequence is finite, i.e. \( H_w (\{c_i\}) = H_w (\{m(x)G(x)\}) < \infty \).
Theorem: A rate \( k/n \) convolutional code with generator matrix \( G(x) \) is not catastrophic if and only if

\[
GCD \left\{ \Delta_i(x) : i = 1, 2, \ldots, \binom{n}{k} \right\} = x^l, \quad l \geq 0,
\]

where \( \Delta_i(x) \) are the determinants of the \( \binom{n}{k} \) distinct \( k \times k \) submatrices of the transfer function matrix \( G(x) \).

Definition: An encoder with generator matrix \( G(x) \) is basic if it is polynomial and has polynomial right inverse (feed forward inverse).

Definition: A minimal encoder is a basic encoder that has the smallest constraint length among all equivalent basic encoders.

Definition: Two generator matrices \( G(x) \) and \( G'(x) \) are equivalent if they generate the same convolutional code \( \{c_i\} \), where \( \{c_i\} \) is the set of all possible output sequences \( c_i \).

Remark: A minimal encoder is not catastrophic.
Consider the following rate $\frac{2}{3}$ encoder:

From the encoder diagram,

$$G(x) = \begin{bmatrix} 1 & x^2 & x \\ x & 1 & 0 \end{bmatrix}. $$
Now, the generator matrix can be written as

\[
G(x) = T(x)G'(x) = \begin{bmatrix}
1 & x^2 \\
x & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & \frac{x}{1+x^3} \\
\frac{x^2}{1+x^3}
\end{bmatrix}.
\]

Furthermore, the encoder output polynomial vector is given by

\[
\tilde{c}(x) = \tilde{m}(x)G(x) = \tilde{m}(x)T(x)G'(x) = \tilde{m}'(x)G'(x),
\]

where \(\tilde{m}'(x) = \tilde{m}(x)T(x)\).

Note that the set of sequences \(\{\tilde{m}(x)\}\) is identical to the set of sequences \(\{\tilde{m}'(x)\}\) if and only if \(T(x)\) is invertible. For the encoder in question,

\[
T^{-1}(x) = \frac{1}{1+x^3} \begin{bmatrix}
1 & x^2 \\
x & 1
\end{bmatrix}.
\]
Hence, the encoder shown next

\[
\begin{pmatrix}
1 & 0 & \frac{x}{1 + x^3} \\
0 & 1 & \frac{x^2}{1 + x^3}
\end{pmatrix},
\]

which is equivalent to the encoder with generator matrix

\[
G(x) = \begin{bmatrix}
1 & x^2 & x \\
x & 1 & 0
\end{bmatrix}.
\]
Theorem: Every code has a systematic encoder that is unique. Furthermore, if $G(x)$ is a polynomial generator matrix, then the generator matrix of the equivalent systematic code is given by

$$G_s(x) = T^{-1}(x)G(x) = \left[ I_k \mid P(x) \right],$$

where $P(x)$ is a $(n-k) \times k$ parity-check matrix with (possibly) rational entries.

Remark: Every systematic encoder for a convolutional code is minimal.

**Encoder State Diagram:**

Let the state of the encoder be described by the contents of the shift registers. For an $(n, k)$ code, each new block of $k$ input bits causes a transition to a new state. If $M = \sum_{i=1}^{k} m_i$, where $m_i$ is the length of the $i$th shift register, is the total encoder memory, then there are a total of $2^M$ states and $2^k$ branches leaving and entering each state.
Example: Consider the following rate $r = \frac{1}{3}$ convolutional encoder, then there are 4 states and 2 branches leave and enter each state, because $k = 1$ and $M = m_1 = 2$. 

\[
\{m_i\} \rightarrow x \rightarrow x \rightarrow \{c_i^1\} \\
\{c_i^2\} \rightarrow x \rightarrow x \rightarrow \{c_i^3\} \\
\{c_i^3\}
\]
The corresponding state diagram is

\[
\begin{align*}
S_0 &\rightarrow 0/000 \\
S_0 &\rightarrow 0/110 \\
S_2 &\rightarrow 0/001 \\
S_2 &\rightarrow 0/111 \\
S_3 &\rightarrow 1/000 \\
S_3 &\rightarrow 1/110 \\
S_1 &\rightarrow 1/001 \\
S_1 &\rightarrow 1/111 \\
S_0 &\rightarrow 0/000 \\
S_1 &\rightarrow 0/110 \\
S_2 &\rightarrow 1/001 \\
S_3 &\rightarrow 1/110 \\
\end{align*}
\]

\[s_0 = (00)\]
\[s_1 = (10)\]
\[s_2 = (01)\]
\[s_3 = (11)\]

**Trellis Diagrams**

They are tree-like structures which are extensions of the convolutional code’s state diagrams that explicitly show the evolution of time.
Example: For the previous rate $r = \frac{1}{3}$ convolutional code, the corresponding trellis, assuming initial state $s_0$, is

For a rate $r = \frac{1}{n}$ convolutional encoder, the following observations can be made:

1. At each time step $t$, there are $2^M$ nodes.
2. There are 2 branches leaving each node, one branch for each input value (0 or 1).
3. After $t = m_d$, there are also 2 branches entering each node.
Suppose a communication system is described by

\[
\begin{align*}
&\text{Convolutional encoder} \quad \Sigma \quad \text{Decoder} \\
&\text{Modulator} \\
&\text{Demodulator} \\
&\text{Decoder}
\end{align*}
\]

**Assumptions:** Channel is memoryless, namely, the noise affects each bit in the received word \( \mathbf{v} \) independently.

**Definition:** A maximum likelihood (ML) decoder selects the estimate \( \hat{\mathbf{c}} \) that maximizes

\[
P\{ \mathbf{v} | \mathbf{c} \}.
\]

**Definition:** A maximum a posteriori (MAP) decoder selects the estimate \( \hat{\mathbf{c}} \) that maximizes

\[
P\{ \mathbf{c} | \mathbf{v} \}.
\]
Let the channel be a memoryless binary symmetric channel (MBSC) with crossover probability \( p = P\{1|0\} = P\{0|1\} \), (same as BER) then \( 1 - p = P\{0|0\} = P\{1|1\} \) and

Let \( P\{c = 0\} = \alpha \), then \( P\{c = 1\} = 1 - \alpha \).
Consider two valid code word sequences $\zeta_1$ and $\zeta_2$, and a received word sequence $\nu$. Each of these sequences is $(t+1)$-bits long, i.e.,

$$\zeta_1 = \begin{bmatrix} c_{10} & c_{11} & c_{12} & \ldots & c_{1t} \end{bmatrix}$$

$$\zeta_2 = \begin{bmatrix} c_{20} & c_{21} & c_{22} & \ldots & c_{2t} \end{bmatrix}$$

$$\nu = \begin{bmatrix} v_0 & v_1 & v_2 & \ldots & v_t \end{bmatrix}$$

Decision Criterion: Let’s apply the MAP criterion, i.e., select $\zeta_1$ if $P\{\nu | \zeta_1\} > P\{\nu | \zeta_2\}$ or $P\{\nu | \zeta_1\} \cdot P\{\zeta_1\} > P\{\nu | \zeta_2\} \cdot P\{\zeta_2\}$. Otherwise, select $\zeta_2$.

Assumption: Let all coded sequences be equiprobable, i.e., $P\{\zeta_i\} = P\{\zeta_j\}$, $\forall \zeta_i, \zeta_j \in C$.

Then, the MAP criterion becomes the ML criterion, which is equivalent to select $\zeta_1$ if $P\{\nu | \zeta_1\} > P\{\nu | \zeta_2\}$, otherwise, select $\zeta_2$. 
Suppose $\zeta_1$ was indeed the transmitted code word sequence, then $\mathbf{v} = \zeta_1 + \mathbf{e}_1$, where $\mathbf{e}_1$ is an error vector. Likewise, if $\zeta_2$ was the transmitted code word sequence, then $\mathbf{v} = \zeta_2 + \mathbf{e}_2$, where $\mathbf{e}_2$ is an error vector. Now, $P\{\mathbf{v} | \zeta_1\} = P\{\zeta_1 + \mathbf{e}_1 | \zeta_1\} = P\{\mathbf{e}_1\}$, because the error probability is independent of the transmitted code word. Let $W_1 \triangleq W_H (\mathbf{e}_1)$ and $W_2 \triangleq W_H (\mathbf{e}_2)$, then the decision criterion becomes

$$P\{\mathbf{e}_1\} = p^{W_1} (1 - p)^{t + 1 - W_1} > p^{W_2} (1 - p)^{t + 1 - W_2} = P\{\mathbf{e}_2\}.$$ 

where $p$ is the average probability of a bit error (BER).

Equivalently,

$$W_1 \ln(p) + (t + 1 - W_1) \ln(1 - p) > W_2 \ln(p) + (t + 1 - W_2) \ln(1 - p)?$$

or $(W_1 - W_2)[\ln(p) - \ln(1 - p)] > 0$?

or $(W_2 - W_1)[\ln(1 - p) - \ln(p)] > 0$?

or $(W_2 - W_1) \ln\left(\frac{1}{p} - 1\right) > 0$?
Again, in practical situations, \( p \ll 0.5 \Rightarrow \ln \left( \frac{1}{p} - 1 \right) > 0 \) and the decision criterion becomes \( W_2 - W_1 > 0 \) or \( W_1 < W_2 \)?

This implies that the ML sequence is the one that has the minimum Hamming distance w.r.t. the assumed path through the trellis.

**Implementation of the ML Decoding Criterion**

Consider a rate \( r = \frac{1}{n} \) encoder whose coded data is transmitted over a MBSC. Let the information sequence be \( L \) bits long. Then the transmitted code word sequence \( c \) contains \( L + M \) blocks, each \( n \)-bit long, i.e.

\[
\zeta = \left( c_0^1 c_0^2 c_0^3 \cdots c_0^n, c_1^1 c_1^2 c_1^3 \cdots c_1^n, \cdots, c_{L+M-1}^1 c_{L+M-1}^2 c_{L+M-1}^3 \cdots c_{L+M-1}^n \right)
\]

since \( M \) memory stages have to be emptied before the next block of \( L \) input information bits arrives at the encoder. Now, the decoder input sequence is

\[
\nu = \left( v_0^1 v_0^2 v_0^3 \cdots v_0^n, v_1^1 v_1^2 v_1^3 \cdots v_1^n, \cdots, v_{L+M-1}^1 v_{L+M-1}^2 v_{L+M-1}^3 \cdots v_{L+M-1}^n \right),
\]
and the conditional probability \( P\{y|c\} \) is given by

\[
P\{y|c\} = P\left\{ v_0^1 v_0^2 \cdots v_0^n | v_1^1 \cdots v_1^n, \ldots, v_{L+M-1}^1 \cdots v_{L+M-1}^n, c \right\} P\left\{ v_1^1 \cdots v_1^n, \ldots, v_{L+M-1}^1 \cdots v_{L+M-1}^n | c \right\}
\]

\[
= P\left\{ v_0^1 \cdots v_0^n | c \right\} P\left\{ v_1^1 \cdots v_1^n | v_2^1 \cdots v_2^n, \ldots, v_{L+M-1}^1 \cdots v_{L+M-1}^n, c \right\} P\left\{ v_2^1 \cdots v_2^n, \ldots, v_{L+M-1}^1 \cdots v_{L+M-1}^n | c \right\}
\]

\[
= P\left\{ v_0^1 \cdots v_0^n | c \right\} P\left\{ v_1^1 \cdots v_1^n | c \right\} \cdots P\left\{ v_{L+M-1}^1 \cdots v_{L+M-1}^n | c_{L+M-1}^1 \cdots c_{L+M-1}^n \right\}
\]

\[
= P\left\{ v_0^1 | c_0^1 \cdots c_0^n \right\} \cdots P\left\{ v_0^n | c_0^1 \cdots c_0^n \right\} \cdots P\left\{ v_{L+M-1}^1 | c_{L+M-1}^1 \cdots c_{L+M-1}^n \right\} \cdots P\left\{ v_{L+M-1}^n | c_{L+M-1}^1 \cdots c_{L+M-1}^n \right\}
\]

\[
= \prod_{i=0}^{L+M-1} \left( \prod_{j=1}^{n} P\left\{ v_i^j | c_i^j \right\} \right)
\]

Equivalently,

\[
\log_x \left[ P\{y|c\} \right] = \sum_{i=0}^{L+M-1} \left( \sum_{j=1}^{n} \log_x \left[ P\left\{ v_i^j | c_i^j \right\} \right] \right).
\]
Let the bit metric be defined by 

\[ M_b \left( v_i^j \mid c_i^j \right) \triangleq a \left[ \log_x \left( P \left( v_i^j \mid c_i^j \right) \right) + b \right], \]

where \( a \) and \( b \) are chosen so that the bit metric is a small positive integer. Then, the path metric of code word \( \zeta \) is defined by

\[ M_p \left( v \mid \zeta \right) \triangleq \sum_{i=0}^{L+M-1} \left( \sum_{j=1}^{n} M_b \left( v_i^j \mid c_i^j \right) \right). \]

The constant \( a \in \mathbb{R}^+ \) is such that the \( \zeta \) that maximizes \( P \left( v \mid \zeta \right) \) also maximizes \( M_p \left( v \mid \zeta \right) \).

A block of \( n \) coded bits corresponds to a single branch in the trellis, thus define the \( i^{\text{th}} \) branch metric by

\[ M_B \left( v_i \mid \zeta_i \right) \triangleq \sum_{j=1}^{n} M_b \left( v_i^j \mid c_i^j \right) \]

and the \( k^{\text{th}} \) partial path metric by

\[ M_p^k \left( v \mid \zeta \right) \triangleq \sum_{i=0}^{k-1} M_B \left( v_i \mid \zeta_i \right) = \sum_{i=0}^{k-1} \left[ \sum_{j=1}^{n} M_b \left( v_i^j \mid c_i^j \right) \right], \quad k \leq L + M \]
Viterbi Decoding Algorithm

Let \( S_{j,t} \triangleq \) Node corresponding to state \( S_j \) at time \( t \).

\[
V(S_{j,t}) \triangleq \text{Accumulated metric at node } S_{j,t}.
\]

1. Set \( V(S_{0,0}) = 0 \) and \( t = 1 \).

2. At time \( t \), compute the practical path metrics for all paths entering each node.

3. Set \( V(S_{k,t}) \) equal to the best partial path metric entering the node corresponding to state \( S_k \) at time \( t \). Ties can be broken in a random fashion. The non-surviving branches are deleted (pruned) from the trellis.

4. If \( t < L + M \), increment \( t \) by one and go to step 2. Otherwise, at time \( t = L + M \), start at state \( S_0 \) and trace the surviving branches backwards through the trellis (the trace-back method).

Once the accumulated metrics have been computed, start at state \( S_0 \) at time \( t = L + M \) and follow the surviving path backward through the trellis. The surviving path is unique and corresponds to the ML code word.
Hard-Decision Decoding

This is equivalent to inputting to the Viterbi decoder the output sequence of a MBSC with crossover probability \( \Pr \{ v_i = 1 \mid c_i = 0 \} = \Pr \{ v_i = 0 \mid c_i = 1 \} = p \) and
\[
\Pr \{ v_i = 1 \mid c_i = 1 \} = \Pr \{ v_i = 0 \mid c_i = 0 \} = 1 - p.
\]

Example: Consider a MBSC with \( 0 < p < \frac{1}{2} \). Let
\[
a = \frac{1}{\log_2 (1-p) - \log_2 (p)}
\]
and
\[
b = -\log_2 (p).
\]
Then
\[
M_{b1} (v_i \mid c_i) = \frac{\log_2 \Pr \{ v_i \mid c_i \} - \log_2 p}{\log_2 (1-p) - \log_2 (p)}
\]
and

<table>
<thead>
<tr>
<th>( M_{b1} (v_i \mid c_i) )</th>
<th>( v_i = 0 )</th>
<th>( v_i = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_i = 0 )</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( c_i = 1 )</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Given these bit metrics, the surviving paths are those paths with the maximum partial path metric at each node. On the other hand, if

\[ a = \frac{1}{\log_2(p) - \log_2(1 - p)} \quad \text{and} \quad b = -\log_2(1 - p), \]

Then

\[ M_{b_2}(v^j_i | c^j_i) = \frac{\log_2 P(v^j_i | c^j_i) - \log_2(1 - p)}{\log_2(p) - \log_2(1 - p)} \]

and

| \( M_{b_2}(v^j_i | c^j_i) \) | \( v^j_i = 0 \) | \( v^j_i = 1 \) |
|--------------------------|-------|-------|
| \( c^j_i = 0 \)          | 0     | 1     |
| \( c^j_i = 1 \)          | 1     | 0     |

implies that the path metric for a code word \( c \) given the received word \( v \) is simply the Hamming distance \( d_H(v, c) \) and the surviving paths are those paths with minimum partial path metric at each node.
Example: Consider the rate \( r = \frac{1}{3} \) convolutional encoder with generator matrix

\[
G(x) = \begin{bmatrix}
1 + x + x^2 & 1 + x + x^2 & 1 + x
\end{bmatrix}.
\]

Then, if the input sequence is \( m = (110101) \), the output coded sequence, after multiplexing, is

\[
\zeta = (111, 000, 001, 001, 111, 001, 111, 110).
\]

Let \( \zeta \) be transmitted through a MBSC and let the output of this channel be given by \( \nu = (101, 100, 001, 011, 111, 101, 111, 110) \), where the bits in error have been underlined. The trellis of this code, along with the path metric labels, assuming the first bit metric, is shown in the next figure.
\[ s_3 \quad \bullet \quad \bullet \quad \bullet \]

\[ s_2 \quad \bullet \quad \bullet \quad \bullet \]

\[ s_1 \quad \bullet \quad \bullet \quad \bullet \]

\[ s_0 \quad \bullet \quad \bullet \quad \bullet \quad \bullet \]

\[ t = 0 \quad 0/000 \quad 1/111 \quad 1/111 \quad 1/111 \]

\[ t = 1 \quad 0/000 \quad 1/111 \quad 1/111 \quad 1/111 \]

\[ t = 2 \quad 0/000 \quad 1/111 \quad 1/111 \quad 1/111 \]

\[ t = 3 \quad 0/000 \quad 1/111 \quad 1/111 \quad 1/111 \]

\[ t = 4 \quad 0/000 \quad 1/111 \quad 1/111 \quad 1/111 \]

\[ t = 5 \quad 0/000 \quad 1/111 \quad 1/111 \quad 1/111 \]

\[ t = 6 \quad 0/000 \quad 1/111 \quad 1/111 \quad 1/111 \]

\[ t = 7 \quad 0/000 \quad 1/111 \quad 1/111 \quad 1/111 \]

\[ t = 8 \quad 0/000 \quad 1/111 \quad 1/111 \quad 1/111 \]

\[ y = \begin{bmatrix} 101 \\ 100 \\ 001 \\ 011 \\ 111 \\ 101 \\ 111 \\ 110 \end{bmatrix} \]
The decoded word is \( \hat{c} = (111, 000, 001, 001, 111, 001, 111, 110) = \bar{c} \)

or,

\[
\hat{m} = (11010100)
\]

to reset encoder.

If we use hard decision decoding when the channel produces burst errors (more than 1 bit error in a block of \( n \) coded bits), then, in general, the ML word obtained by the Viterbi decoder also contains most of the errors caused by the error event.

**Soft-Decision Decoding**

Let the convolutionally coded sequence \( c \) be BPSK modulated and transmitted over a memoryless AWGN channel with power spectral density \( S_N(f) = \frac{N_0}{2} \), \( \forall f \). Furthermore, let the Gaussian noise be zero-mean, i.e., \( f_N(n) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{n^2}{N_0}} \).
Finally, let the transmitted code word average bit energy be $E_b$ and the coded bits $\{c_i^j\}$ take on the values of $\pm 1$, i.e.,

$$f(v_j^i|c_i^j) = \frac{1}{\sqrt{\pi N_0}} \exp \left[ - \frac{(v_j^i - c_i^j \sqrt{E_b})^2}{N_0} \right].$$
Let the bit likelihood function be \( f(v_i^j | c_i^j) \) then the log likelihood function is given by

\[
\ln \left[ f(v|\zeta) \right] = \sum_{i=0}^{L+M-1} \left[ \sum_{j=1}^{n} \ln \left[ f(v_i^j | c_i^j) \right] \right]
\]

\[
= \sum_{i=0}^{L+M-1} \left[ \sum_{j=1}^{n} \left( -\frac{(v_i^j - c_i^j \sqrt{E_b})^2}{N_0} - \ln \sqrt{\pi N_0} \right) \right]
\]

\[
= -\frac{1}{N_0} \sum_{i=0}^{L+M-1} \left[ \sum_{j=1}^{n} \left[ (v_i^j)^2 - 2\sqrt{E_b} v_i^j c_i^j + E_b \right] \right] - \frac{(L+M)n}{2} \ln (\pi N_0)
\]

\[
= \frac{2\sqrt{E_b}}{N_0} \sum_{i=0}^{L+M-1} \sum_{j=1}^{n} v_i^j c_i^j - \frac{1}{N_0} \sum_{i=0}^{L+M-1} \sum_{j=1}^{n} (v_i^j)^2 - \frac{(L+M)n}{2} \ln (\pi N_0)
\]

\[
= k_1 v \cdot \zeta + k_2,
\]
where $\bar{v} \cdot \bar{c} = \sum_{i=0}^{L+M-1} \sum_{j=1}^{n} v_i^j c_i^j$, $k_1 = \frac{2\sqrt{E_b}}{N_0}$, and

$$k_2 = -\frac{1}{N_0} \sum_{i=0}^{L+M-1} \sum_{j=1}^{n} \left[ (v_i^j)^2 + E_b \right] - \frac{(L+M)n}{2} \ln(\pi N_0).$$

Since $k_1 > 0$ and the addition of a constant quantity does not change the maximization procedure, the equivalent path and bit metrics for the AWGN channel are

$$M_p (v | \bar{c}) = \bar{v} \cdot \bar{c} \quad \text{and} \quad M_b \left( v_i^j | c_i^j \right) = v_i^j c_i^j,$$

respectively. So, the goal is to maximize the dot product (alignment) of the vectors $\bar{v}$ and $\bar{c}$. This is equivalent to the minimization of the distance $\|v - \bar{c}\|^2$!

Practical implementation of the soft Viterbi decoding algorithm often requires the A/D conversion of the output of the detector. However, a relatively small number of quantization levels is generally required to get good results.
Example: Let the encoded data of the rate \( r = \frac{1}{3} \) convolutional encoder of the previous example be BPSK modulated and transmitted over a MAWGN channel. Let the received data be A/D by a 2-bit ADC (4 quantization levels) before decoding it by a Viterbi decoder, i.e.

\[
\hat{v} \quad \bar{1} \\
\bar{0} \\
0
\]

where \( \bar{0} \triangleq \) strong zero and \( \bar{1} \triangleq \) strong 1.
Let the bit (symbol) metric be $M \{ \hat{v}_i^j | c_i^j \} = \frac{3}{2} \left[ \log_2 \left( P \{ \hat{v}_i^j | c_i^j \} \right) - \log_2 (0.05) \right]$, then

| $P \{ \hat{v}_i^j | c_i^j \}$ | $\hat{v}_i^j$ = \ \begin{array}{cccc}
0 & 0 & 1 & \bar{1} \\
\end{array} |
|-----------------------------|-----------------|
| $c_i^j = 0$ | \begin{array}{cccc}
0.5 & 0.32 & 0.13 & 0.05 \\
\end{array} |
| $c_i^j = 1$ | \begin{array}{cccc}
0.05 & 0.13 & 0.32 & 0.5 \\
\end{array} |

| $M \{ \hat{v}_i^j | c_i^j \}$ | $\hat{v}_i^j$ = \ \begin{array}{cccc}
0 & 0 & 1 & \bar{1} \\
\end{array} |
|-----------------------------|-----------------|
| $c_i^j = 0$ | \begin{array}{cccc}
5 & 4 & 2 & 0 \\
\end{array} |
| $c_i^j = 1$ | \begin{array}{cccc}
0 & 2 & 4 & 5 \\
\end{array} |
Assuming the same input message sequence to the encoder as in the last example, let the received sequence be given by

$$\hat{y} = (101, 100, 001, \bar{0}11, \bar{1}10, \bar{1}10, \bar{1}11, 1\bar{1}0).$$

Let’s find the decoded sequence using the Viterbi algorithm.
The decoded sequence is therefore
\[
\hat{c} = (111, 000, 001, 111, 001, 111, 110) = \zeta \text{ or } \hat{m} = (11010100) = m.
\]
Since \( \hat{c} = \zeta \), we can conclude that the added soft-decision information has enabled the Viterbi decoder to correct the error burst.

**Viterbi Decoder Practical Issues**
- The decoder generally needs to output decoded information bits before the entire encoded message has been received.
- Incoming signals must be quantized by an A/D converter.
- A form of block synchronization is necessary, since the decoder does not know when an n-bit block ends and the next begins when it is first turned on.

The trace back method of Viterbi decoding (R. B. Wells, Applied coding and information theory for engineers, section 6.7, 1999, Prentice Hall) is the most practical and implementable version of the algorithm.
Performance Analysis
The state diagram of a code can be modified to obtain information about the performance of a convolutional code.

Let a branch connecting two states have gain $\delta^l$, where $l$ is the Hamming weight of the $n$ convolutionally encoded bits on that branch. Let $S_0$ be the initial and final state, and delete the self loop around it. Then each path that connects the initial and the final state represents a nonzero code word that diverges from and remerges with state $S_0$ exactly once.

Example: The state diagram and the modified state diagram of the rate $\frac{1}{2}$ encoder of the first example are shown in the next two figures
State diagram

\[ \delta_0 = (0,0) \]
\[ \delta_1 = (1,0) \]
\[ \delta_2 = (0,1) \]
\[ \delta_3 = (1,1) \]
Modified state diagram
Let $T(\delta)$ be the generating function (transfer function) of the code. Then, using block diagram algebra (assuming positive feedback for the loops), we get

$$T(\delta) = \frac{\delta^5}{1-2\delta} = \delta^5 + 2\delta^6 + 4\delta^7 + \cdots + 2^i \delta^{i+5} + \cdots,$$

which implies that the code contains 1 nonzero code word of weight 5, 2 code words of weight 6, 4 of weight 7, and, in general, $2^i$ code words of weight $i+5$, $i = 0, 1, 2, \cdots$.

Example: The state diagram and the modified state diagram of the (3, 2) encoder we previously considered are shown below.
State diagram
Modified state diagram
Again, using block diagram algebra, one can verify that the transfer function is given by

\[ T(\delta) = \frac{2\delta^3 + \delta^4 + \delta^5 + \delta^6 - \delta^7 - \delta^8}{1 - 2\delta - 2\delta^2 - \delta^3 + \delta^4 + \delta^5} = 2\delta^3 + 5\delta^4 + 15\delta^5 + \cdots, \]

which implies that the code contains 2 nonzero code words of weight 3, 5 of weight 4, 15 of weight 5, and so on.

Additional information about the structure of the convolutional code can be obtained by relabeling the branches of the modified state diagram.

Let the exponents of \( \beta \) and \( \lambda \) indicate the weight of the input information block of \( k \) bits and the normalized time duration of a branch traversal, respectively. Then we can obtain the modified transfer function

\[ T(\delta, \beta, \lambda) = \sum_{i} \sum_{j} \sum_{l} A_{i,j,l} \delta^i \beta^j \lambda^l, \]

where \( A_{i,j,l} \) is the number of code words with weight \( i \), associated with the information sequence with weight \( j \), and whose length is \( l \) branches.
Example: The newly modified state diagram of the (2, 1) encoder we have been working with is

For this code, the modified transfer function is given by

$$T(\delta, \beta, \lambda) = \frac{\delta^5 \beta \lambda^3}{1 - \delta \beta \lambda (1 + \lambda)} = \delta^5 \beta \lambda^3 + \delta^6 \beta^2 \lambda^4 (1 + \lambda) + \cdots + \delta^{i+5} \beta^{i+1} \lambda^{i+3} (1 + \lambda)^i + \cdots.$$  

When $\lambda = 1$, we get

$$T(\delta, \beta, \lambda) \bigg|_{\lambda = 1} = \delta^5 \beta + 2 \delta^6 \beta^2 + 2 \delta^7 \beta^3 + \cdots + 2^i \delta^{i+5} \beta^{i+1} + \cdots.$$
Although the generating function of the code provides substantial information about it, it is often difficult to compute in practice whenever the constraint length $v$ is larger than 5. This means that other mechanisms have to be found to estimate the performance of a given convolutional code.

The performance properties of a convolutional code depend on the decoding algorithm and the distance properties of the code itself. 

**Definition:** The minimum free distance $d_{\text{free}}$ of the code is the minimum Hamming distance between all pairs of complete convolutional code words, i.e.

$$d_{\text{free}} \triangleq \min_{i, j \neq i} \{d\left(c_i, c_j\right) : c_i \neq c_j\}.$$ 

where $d\left(c_i, c_j\right)$ is the Hamming distance between the code words $c_i$ and $c_j$.

Equivalently,

$$d_{\text{free}} = \min \{H_w(mG) : m \neq 0\}.$$ 

Qualitatively, it is the minimum Hamming weight of all paths in the state diagram that diverge from and remerge with the zero state $S_0$. 

179
Example: The (2, 1) encoder with generator matrix \( G(x) = \begin{bmatrix} 1 + x + x^2 & 1 + x^2 \end{bmatrix} \) has the generating (transfer) function

\[
T(\delta) = \frac{\delta^5}{1 - 2\delta} = \delta^5 + 2\delta^6 + 4\delta^7 + \cdots + 2^i \delta^{i+5} + \cdots.
\]

This means that \( d_{\text{free}} = 5 \) (the exponent of the first term in the series).

Let the truncated input and output sequences of the encoder be

\[
m_i = (m_0^{(1)} \cdots m_0^{(k)}, m_1^{(1)} \cdots m_1^{(k)}, \ldots, m_i^{(1)} \cdots m_i^{(k)})
\]

and

\[
c_i = (c_0^{(1)} \cdots c_0^{(n)}, c_1^{(1)} \cdots c_1^{(n)}, \ldots, c_i^{(1)} \cdots c_i^{(n)}).
\]

Define the column distance function of the code by

\[
d_{c_i} \equiv \min_{m_i \neq 0} \{ H_w (c_i) \}, \quad i = 0, 1, 2, \ldots.
\]
Example: Consider the (2, 1) encoder of the first example. In this case, we are going to generate $d_{ci}$ by injecting various input sequences into the encoder and measure the weight of the output. The following figure shows a plot of $d_{ci}$ vs. $i$ for the (2, 1) convolutional code with unit sample responses $g^{(1)} = (111)$ and $g^{(2)} = (101)$.
Remark: Free distance is the fundamental limitation on convolutional code performance. Also, it is the main figure of merit when maximum likelihood sequence decoding is used. Moreover, when the constraint length \( v \) is large, \( d_{\text{free}} \) is not easy to compute and estimates of it can be used to assess code performance. In fact,

\[
d_{\text{free}} \leq vn = (m_d + 1)n.
\]

The justification for this bound is that an input information (message) sequence with a single nonzero vector \( m_0 \neq 0 \) can produce at most \( (m_d + 1)n \) coded bits that are nonzero due to the encoder’s \( M \) memory shifts.

Example: Our \((2, 1)\) encoder has \( v = 3 \) and \( n = 2 \). Thus, \( d_{\text{free}} \leq 6 \). In fact, \( d_{\text{free}} = 5 \).
A tighter and more general upper bound is

\[ d_{\text{free}} \leq \min_{i, i=1,2,\ldots} \left\{ \left\lfloor \frac{q^{ik-1}}{q^k - 1} (i + m_q) n(q - 1) \right\rfloor \right\}, \]

where \( \lfloor \cdot \rfloor \) is the floor function, \( i \) is the length of the input sequence \( m \), and \( q \) stands for \( q \)-ary codes (\( q = 2 \) implies binary).

Similar to block codes, the free distance of convolutional codes is very important in the sense that it provides information about the error correcting capability of the code. Specifically, the number of errors which can be corrected by a convolutional code is

\[ t = \left\lfloor \frac{d_{\text{free}} - 1}{2} \right\rfloor. \]
Simple BER Performance Upper Bounds:

Simple BER performance upper bounds of binary convolutional codes with rate $k/n$ can be obtained using the transfer function of the code. For transmission over a BSC and maximum likelihood decoding, the BER upper bound is described by

$$p_b = BER < \left( \frac{1}{k} \right) \left. \frac{\partial T(\delta, \beta, \lambda)}{\partial \beta} \right|_{\delta = \sqrt{4pq}, \beta = 1, \lambda = 1}.$$

For binary transmission over an AWGN channel with PSD $S_W(f) = \frac{N_0}{2}$, $\forall f$, the BER upper bound is described by

$$p_b = BER < \left( \frac{1}{k} \right) \left. \frac{\partial T(\delta, \beta, \lambda)}{\partial \beta} \right|_{\delta = e^{-\frac{E_b}{2N_0}}, \beta = 1, \lambda = 1},$$

where $\frac{E_b}{N_0}$ is the average SNR and $R = \frac{k}{n}$ is the code rate.
For sufficiently high average SNR (large $\frac{E_b}{N_0}$ and small $p$), tighter bounds can be obtained.

For transmission over a BSC and maximum likelihood decoding, the BER upper bound is described by

$$p_b = BER < \left( \frac{1}{k} \right) \left[ \frac{\partial T(\delta, \beta, \lambda)}{\partial \beta} \right] \left[ \frac{\partial T(-\delta, \beta, \lambda)}{\partial \beta} \right]$$

For binary transmission over an AWGN channel with PSD $S_w(f) = \frac{N_0}{2}$, $\forall f$, the BER upper bound is described by

$$p_b = BER < \left( \frac{1}{k} \right) Q \left( \sqrt{2Rd_f \frac{E_b}{N_0}} \right) e^{2Rd_f \frac{E_b}{N_0} \frac{\partial T(\delta, \beta, \lambda)}{\partial \beta}}$$

where $d_f$ is the free distance of the code.
In terms of performance, nonsystematic codes are superior than systematic codes because of their larger free distance $d_{\text{free}}$. This is manifested by the fact that the decoding process generally exceeds the encoder’s constraint length.

The following tables (S. Lin and D. J. Costello, Jr., Error control coding: Fundamentals and applications, Prentice Hall, 1983) list several encoders with optimal (largest) $d_{\text{free}}$. Note that the encoders descriptions are given in octal notation, e.g., 5 in octal notation is equal to 101 in binary notation and 561 in octal notation is equal to 101110001 in binary notation, etc..
Rate \( \frac{1}{2} \) convolutional codes with maximal minimum free distance

<table>
<thead>
<tr>
<th>( \nu )</th>
<th>( g^{(1)} )</th>
<th>( g^{(2)} )</th>
<th>( d_{\text{free}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>74</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>46</td>
<td>72</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>65</td>
<td>57</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>554</td>
<td>744</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>712</td>
<td>476</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>561</td>
<td>753</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>4734</td>
<td>6624</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>4672</td>
<td>7542</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>4335</td>
<td>5723</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>42554</td>
<td>77304</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td>43572</td>
<td>56246</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>56721</td>
<td>61713</td>
<td>18</td>
</tr>
<tr>
<td>16</td>
<td>447254</td>
<td>627324</td>
<td>19</td>
</tr>
<tr>
<td>17</td>
<td>716502</td>
<td>514576</td>
<td>20</td>
</tr>
</tbody>
</table>
Rate 1/3 convolutional codes with maximal minimum free distance

<table>
<thead>
<tr>
<th>$\nu$</th>
<th>$g^{(1)}$</th>
<th>$g^{(2)}$</th>
<th>$g^{(3)}$</th>
<th>$d_{\text{free}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>54</td>
<td>64</td>
<td>74</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>66</td>
<td>76</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>47</td>
<td>53</td>
<td>75</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>554</td>
<td>624</td>
<td>764</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>452</td>
<td>662</td>
<td>756</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>557</td>
<td>663</td>
<td>711</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>4474</td>
<td>5724</td>
<td>7154</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>4726</td>
<td>5562</td>
<td>6372</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>4767</td>
<td>5723</td>
<td>6265</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>42554</td>
<td>43364</td>
<td>77304</td>
<td>24</td>
</tr>
<tr>
<td>14</td>
<td>43512</td>
<td>73542</td>
<td>76266</td>
<td>26</td>
</tr>
</tbody>
</table>
Rate 2/3 convolutional codes with maximal minimum free distance

<table>
<thead>
<tr>
<th>$v$</th>
<th>$M$</th>
<th>$g_1^{(1)}$</th>
<th>$g_2^{(1)}$</th>
<th>$g_1^{(2)}$</th>
<th>$g_2^{(2)}$</th>
<th>$g_1^{(3)}$</th>
<th>$g_2^{(3)}$</th>
<th>$d_{\text{free}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>60</td>
<td>14</td>
<td>30</td>
<td>40</td>
<td>70</td>
<td>74</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>64</td>
<td>30</td>
<td>64</td>
<td>74</td>
<td>64</td>
<td>74</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>60</td>
<td>16</td>
<td>34</td>
<td>46</td>
<td>54</td>
<td>74</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>64</td>
<td>26</td>
<td>12</td>
<td>66</td>
<td>52</td>
<td>44</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>52</td>
<td>06</td>
<td>06</td>
<td>70</td>
<td>74</td>
<td>53</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>63</td>
<td>15</td>
<td>46</td>
<td>65</td>
<td>61</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
Rate $\frac{3}{4}$ convolutional codes with maximal minimum free distance

| $v$ | $M$ | $g_1^{(1)}$ | $g_1^{(2)}$ | $g_1^{(3)}$ | $g_1^{(4)}$ | $g_2^{(1)}$ | $g_2^{(2)}$ | $g_2^{(3)}$ | $g_2^{(4)}$ | $g_3^{(1)}$ | $g_3^{(2)}$ | $g_3^{(3)}$ | $g_3^{(4)}$ | $d_{free}$ |
|-----|-----|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|----------|
| 2   | 3   | 4           | 4           | 4           | 4           | 2           | 6           | 2           | 5           | 5           | 0           | 6           | 2           | 5           | 4         |
| 3   | 5   | 6           | 1           | 2           | 0           | 2           | 6           | 0           | 6           | 7           | 0           | 7           | 4           | 5           | 5         |
| 3   | 6   | 6           | 4           | 1           | 4           | 0           | 7           | 0           | 6           | 7           | 7           | 4           | 4           | 4         |
| 4   | 8   | 70          | 14          | 40          | 30          | 20          | 50          | 14          | 74          | 20          | 40          | 74          | 40          | 7         |
| 4   | 9   | 40          | 04          | 04          | 14          | 00          | 34          | 20          | 60          | 34          | 60          | 20          | 64          | 8         |
Remark: Nonsystematic convolutional codes offer a higher minimum free distance than systematic codes of comparable constraint length and rate, e.g., for rate $\frac{1}{3}$ convolutional encoders with $\nu = 7$,

$$\max \{d_{\text{free}}\} = 12 \text{ for systematic codes}$$

$$\max \{d_{\text{free}}\} = 15 \text{ for nonsystematic codes}$$

Suppose that a long sequence of 0’s has been transmitted through the channel. An error event occurs when at node $S_{0,t}$ a nonzero path leaves that node and, after remerging with the all-zero path at a later node, is declared the survivor.
Example: Consider a 4 state $r = \frac{1}{n}$ convolutional encoder. The trellis diagram with the ML path as well another path are shown in the next figure. Note that it is assumed that an all-zero sequence was transmitted through the channel and the two paths constitute error events.
Definition: The node error probability $P_{e} \left( S_{0,j} \right)$ at node $S_{0,j}$ is the probability that a nonzero path leaving node $S_{0,j}$ accumulates a higher partial path metric than the all-zero path before remerging with the all-zero path.

Let $E' \left( S_{0,j} \right) \triangleq \{ \epsilon' \}$, the set of all nonzero paths diverging from node $S_{0,j}$.

$\Delta M \left( \epsilon', 0 \right) \triangleq$ The amount by which the partial path metric of the all-zero path is exceeded by that of $\epsilon'$, when $\epsilon'$ remerges with the all-zero path $0$.

Then the union-bound estimate says that

$$P_{e} \left( S_{0,j} \right) \leq P \left\{ \bigcup_{\epsilon' \in E' \left( S_{0,j} \right)} \{ \Delta M \left( \epsilon', 0 \right) \geq 0 \} \right\} \leq \sum_{\epsilon' \in E' \left( S_{0,j} \right)} P \{ \Delta M \left( \epsilon', 0 \right) \geq 0 \}.$$
Consider now the transmission over a MBSC with crossover probability $p$ and hard decision Viterbi decoding.

Regardless of the channel used for transmission, $P_E(j)$, the probability of selecting an erroneous path at the $j^{th}$ node, given that the all-zero sequence has been transmitted, is upper-bounded by

$$P_E(j) \leq \sum_{\ell=1}^{\infty} a_\ell P_\ell,$$

where $P_\ell \triangleq P\left\{\text{pair-wise error in favor of an incorrect path that differs in } \ell \text{ symbols from the correct (all-zeros) path over the unmerged span}\right\}$. $a_\ell \triangleq \text{number of nonzero paths with weight } \ell \text{ leaving the all-zero state.}$

$P_E \triangleq P_e(S_{0,j}) = P_e \triangleq \text{Node error probability.}$
Let \( b_\ell \) be the total number of nonzero information bits associated with code words of weight \( \ell \). Then, for a rate \( r = \frac{1}{n} \) convolutional encoder the BER \( (p_b) \) is bounded from above by

\[
p_b \leq \sum_{\ell=1}^{\infty} b_\ell P_\ell.
\]

Moreover, \( b_\ell = \sum_{s=1}^{\infty} s a_{\ell,s}, \)

where \( a_{\ell,s} \) is the number of code words of weight \( \ell \) associated with information sequences of weight \( s \).

The probability \( P_\ell \) depends on the type of channel used for transmission of the coded sequence.

So, for the MBSC, let \( \bar{x} \) be the all-zero sequence and \( \bar{y} \) a valid code word such that

\[
d_H(\bar{x}, \bar{y}) = \ell = W_H(y + 0) = W_H(y)
\]
Let $\chi$ be the transmitted sequence and let the received sequence be $\psi$, then

$$d_H(\psi, \chi) = d_H(\psi, 0) = d_H(\psi + \psi + \psi, 0) = W_H(\psi + \psi + \psi + 0)$$

$$\leq W_H(\psi + \psi) + W_H(\psi) = d_H(\psi, \psi) + \ell$$

implies that

$$d_H(\psi, \chi) - d_H(\psi, \psi) \leq \ell$$

and

$$d_H(\chi + \epsilon_x, \chi) - d_H(\chi + \epsilon_y, \chi) = d_H(\epsilon_x, 0) - d_H(\epsilon_y, 0) \leq \ell$$

and

$$W_H(\epsilon_x) - W_H(\epsilon_y) \leq \ell.$$ 

An error will be made by the decoder if $W_H(\epsilon_x) > W_H(\epsilon_y)$ and $W_H(\epsilon_x) > \frac{\ell}{2}$. 

196
Moreover, if $W_H(e_x) > \ell$, $e_x(i) = e_y(i) = "1"$ and such bits do not contribute when the path metrics are compared. Therefore, $\frac{\ell}{2} < W_H(e_x) \leq \ell$ and

$$P_\ell = \sum_{k=\ell+1}^{\ell} p^k (1-p)^{\ell-k}, \ell \text{ odd}$$

or

$$P_\ell = \sum_{k=\ell+1}^{\ell} p^k (1-p)^{\ell-k} + \frac{1}{2} \ell \left( \frac{\ell}{2} \right) p^{\ell/2} (1-p)^{\ell/2}, \ell \text{ even}$$

Suppose now that the convolutionally coded sequence is BPSK modulated and transmitted over a MAWGN channel with power spectral density $S_N(f) = \frac{N_0}{2}$, $\forall f$.

Suppose further that the transmitted sequence is the all-1's sequence (the all-zeros coded sequence). Then $P_\ell$ is the probability that another path in the trellis with $\ell$ bits equal to 1 is more positive than negative (assuming the inner product metric), i.e.,

$$P_\ell = P\left\{ \sum_{k=1}^{\ell} v_i^j \geq 0 \right\} \text{ for some } i, j,$$
where $^{k}v_{i}^{j} \sim G\left(-\sqrt{E_{b}}, \frac{N_{0}}{2}\right)$ when a sequence of -1’s has been transmitted. In other words,

$$f\left(^{k}v_{i}^{j} \mid ^{k}c_{i}^{j} = -1\right) = \frac{1}{\sqrt{\pi N_{0}}} e^{-\frac{(^{k}v_{i}^{j})^2}{N_{0}}}$$

and

$$\sum_{k=1}^{f}^{k}v_{i}^{j} \sim G\left(-l\sqrt{E_{b}}, l \frac{N_{0}}{2}\right),$$

which implies that

$$P_{\ell} = \frac{1}{\sqrt{\pi \ell N_{0}}} \int_{0}^{\infty} e^{-\frac{(u+\ell\sqrt{E_{b}})^2}{2\ell N_{0}}} du = Q\left(\sqrt{\frac{2\ell E_{b}}{N_{0}}}\right),$$

where $E_{b}$ is the received coded bit energy.
From either the state diagram or the trellis of the rate $r = \frac{1}{3}$ convolutional encoder in question, $\ell_{\min} = 6$ and $a_{\ell_{\min}} = 1$

Hence, $P_e = P_e\left(S_{0,j}\right) = P_E\left(j\right) \leq \sum_{\ell=6}^{\infty} a_{\ell} P_{\ell}$ and $p_b \leq \sum_{\ell=6}^{\infty} b_{\ell} P_{\ell}$.

Let $d_{\text{free}} = \ell_{\min}$, then for any rate $r = \frac{1}{n}$ convolutional code,

$$P_e \leq \sum_{\ell = d_{\text{free}}}^{\infty} a_{\ell} P_{\ell} \quad \text{and} \quad p_b \leq \sum_{\ell = d_{\text{free}}}^{\infty} b_{\ell} P_{\ell}.$$ 

Tables 6.5.1-6 in [R. B. Wells, Applied coding and information theory for engineers, Prentice Hall, 1999] provide good and proven convolutional codes and their performances evaluation coefficients.

Example: Calculate the BER performance of a $r = \frac{1}{2}$ encoder with generator polynomial matrix $(23, 35)_8$ and $d_{\text{free}} = 7$. From table 6.5.2,

$$p_b \leq \sum_{j=0}^{\infty} n_{j} P_{j+d_{\text{free}}} = \sum_{\ell = d_{\text{free}}}^{\infty} n_{\ell-d_{\text{free}}} P_{\ell} = \sum_{\ell = d_{\text{free}}}^{\infty} b_{\ell} P_{\ell} = \sum_{\ell = 7}^{\infty} b_{\ell} P_{\ell}.$$
Thus, \( b_\ell = n_{\ell - d_{\text{free}}} = n_{\ell - 7}, \ell = 7, 8, \ldots, 14, \ldots \).

The first 8 terms of the sum are given by

\[
\sum_{\ell=7}^{14} b_\ell P_\ell = 4P_7 + 12P_8 + 20P_9 + 72P_{10} + 255P_{11} + 500P_{12} + 1324P_{13} + 3680P_{14}
\]

If the channel is AWGN and soft Viterbi decoding is used, then with \( \text{SNR} = \frac{E_b}{N_0} \),

\[
\sum_{\ell=7}^{14} b_\ell P_\ell = 4Q\left( \sqrt{ \frac{14 E_b}{N_0} } \right) + 12Q\left( \sqrt{ \frac{16 E_b}{N_0} } \right) + \cdots + 3680Q\left( \sqrt{ \frac{28 E_b}{N_0} } \right).
\]

Further Performance Bounds

Let the input message sequence \( m \) have the length \( N \). Then a rate \( r = \frac{1}{n} \) convolutional code with constraint length \( \nu \) may be viewed as a block code with \( 2^N \) code words of length \( n(N + \nu) \).
The BER performance of a finite-length convolutional code with ML decoding on an AWGN channel with \(SNR = \frac{E_b}{N_0}\) is bounded above by

\[
p_b \leq \sum_{l=d_{\text{free}}}^{n(N+y)} \frac{N_l \cdot \tilde{W}_l}{N} Q\left(\sqrt{\frac{lrE_b}{2N_0}}\right), \quad \tilde{W}_l = \frac{W_l}{N_l},
\]

where \(E_b = \) information bit energy, \(N_l =\) number of code words with weight \(l\), and \(W_l =\) total information weight of all code words of weight \(l\). (Information weight = Hamming weight.)

**Punctured Convolutional Codes**

Puncturing of a convolutional code is accomplished by a periodic deletion of bits from one or more of the encoder output streams. Then net effect of this operation is a higher-rate code.
Example: Consider the following rate $\frac{1}{2}$ convolutional encoder

\[ \zeta = \left( c_0^{(1)} c_0^{(2)}, c_1^{(1)} c_1^{(2)}, c_2^{(1)} c_2^{(2)}, \ldots, c_i^{(1)} c_i^{(2)}, \ldots \right) \]
If every 4\textsuperscript{th} bit is deleted from $c$ before transmission, then

$$
\zeta_p = \left(c_0^{(1)} c_0^{(2)}, c_1^{(1)} -, c_2^{(1)} c_2^{(2)}, c_3^{(1)} -, \ldots \right).
$$

This means that three coded bits are generated for every two input information bits. By doing so, a rate $\frac{2}{3}$ code can be generated.

If the receiver inserts erasures at the points where the bits were deleted, the decoder for the convolutional code $c$ can be used to decode the punctured code $c_p$. From the implementation point of view, such erasures pose a small timing problem, since one output bit is generated at twice the rate of the other output bit.

Let the input information bit sequence to the above rate $\frac{1}{2}$ convolutional encoder be $m=(1101001110\ldots)$. 
Also, let the initial state of the encoder be $S_0 = (0 \ 0 \ )$. The following table shows the actions of the encoder.

<table>
<thead>
<tr>
<th>time step $i$</th>
<th>input</th>
<th>state</th>
<th>output</th>
<th>next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>00</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>10</td>
<td>0 1</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>11</td>
<td>0 1</td>
<td>01</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>01</td>
<td>0 0</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>01</td>
<td>1 1</td>
<td>00</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>00</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>10</td>
<td>0 1</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>11</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>11</td>
<td>0 1</td>
<td>01</td>
</tr>
</tbody>
</table>

Then

$\zeta = (11, 01, 01, 00, 10, 11, 11, 01, 10, 01, \ldots)$ and $\zeta_p = (110, 010, 101, 110, 100, \ldots)$
Let us now examine the following rate $\frac{2}{3}$ convolutional encoder.

Let $m^{(1)}$ represent the odd component of $\sim m$ and $m^{(2)}$ represent the even components of $\sim m$. 
Then the output sequence of the rate $\frac{2}{3}$ convolutional encoder is

$$c_{2/3} = \left( c_0^{(1)} c_0^{(2)} c_0^{(3)}, c_1^{(1)} c_1^{(2)} c_1^{(3)}, c_2^{(1)} c_2^{(2)} c_2^{(3)}, c_3^{(1)} c_3^{(2)} c_3^{(3)}, \ldots \right)$$

and can be obtained as follows:

<table>
<thead>
<tr>
<th>time step $i$</th>
<th>$m_i^{(1)} m_i^{(2)}$</th>
<th>state</th>
<th>output</th>
<th>next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>00</td>
<td>110</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
<td>11</td>
<td>010</td>
<td>01</td>
</tr>
<tr>
<td>3</td>
<td>00</td>
<td>01</td>
<td>101</td>
<td>00</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>00</td>
<td>110</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>11</td>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>

$\therefore c_{2/3} = (110, 010, 101, 110, 100, \ldots) = c_p$!
Let us analyze the two encoders. We can draw the trellis of the rate $\frac{1}{2}$ encoder using the information found in the corresponding state diagram, that is,
The state diagram of the rate \( \frac{2}{3} \) encoder can be constructed from the following table.

<table>
<thead>
<tr>
<th>( m_i^{(1)} )</th>
<th>( m_i^{(2)} )</th>
<th>state</th>
<th>output</th>
<th>next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td></td>
<td>0 0</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td></td>
<td>0 0</td>
<td>0 0</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>1 1</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>0 0</td>
<td>0 0</td>
<td>1 0</td>
<td>1</td>
</tr>
<tr>
<td>0 0</td>
<td></td>
<td>0 1</td>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td></td>
<td>0 1</td>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td></td>
<td>0 1</td>
<td>0 1</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td></td>
<td>0 1</td>
<td>0 1</td>
<td>1</td>
</tr>
</tbody>
</table>
The trellis diagram below (as well as the state diagram) show that transitions to any of the four states can occur. This implies that an ML-type decoder would have to perform three comparisons per received block of coded bits.

A ML decoder for the punctured code based on the rate $\frac{1}{2}$ encoder, on the other hand, would only have to perform two comparisons per received block of coded bits! This is indeed a computational advantage.
Remark: Not every code $c_p$ resulting from puncturing a good rate $\frac{1}{2}$ encoder is a “good” code. A list of “good” punctured convolutional codes can be found in: J. B. Cain, G. C. Clark, and J. M. Geist. “Punctured Convolutional Codes of rate $(n-1)/n$ and Simplified Maximum Likelihood Decoding,” IEEE Trans. on Information Theory, IT-25, pp 97-100, 1979.