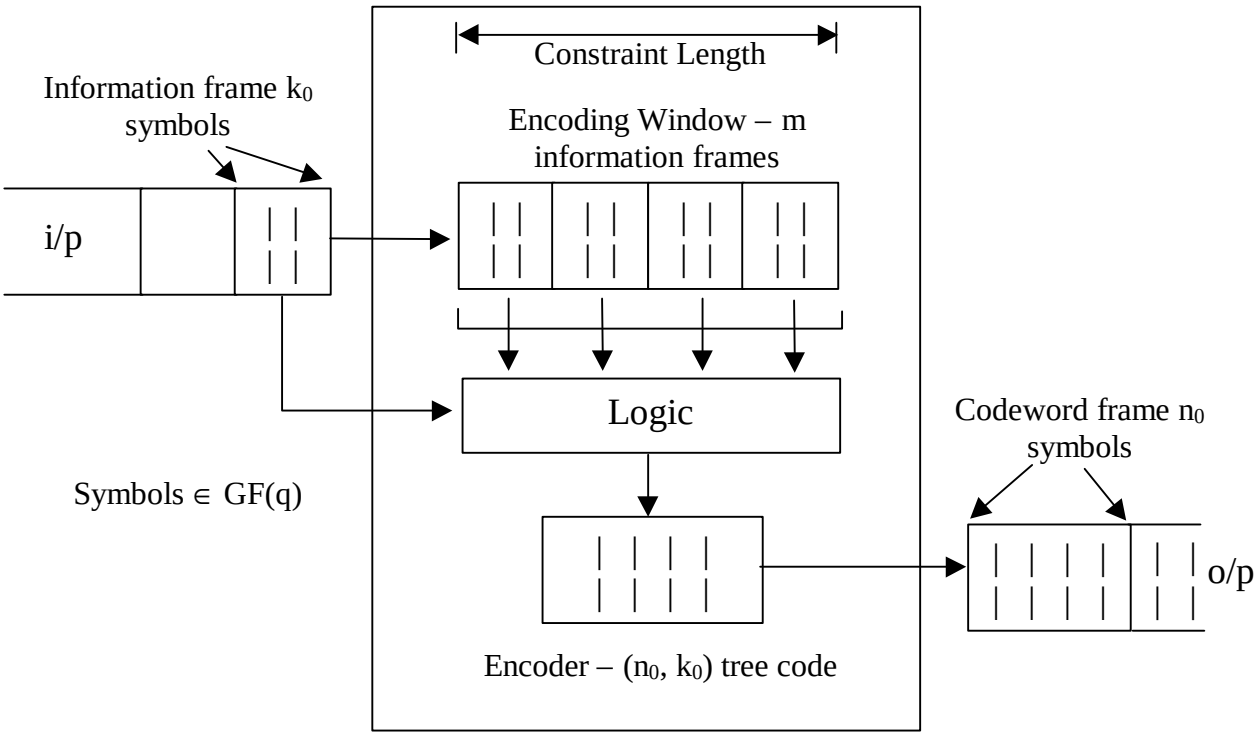
Tree Codes



Rate $R \equiv \frac{k_0}{n_0}$; Constraint length $v \equiv mk_0$

Word length $k = (m+1)k_0$; Block length $n \equiv (m+1)n_0$

$$n = k \cdot \frac{n_0}{k_0} = \frac{k}{R}$$

Definition : Tree Code – (n_0, k_0) : Mapping from a set of semi-infinite sequences of elements of GF(q) into itself such that if for any m, 2 semiinfinite sequences agree in the 1^{st} mk_0 components then their image agree in the 1^{st} mn_0 components.

Properties :

a) Finite Constraint Length – $m < \infty$. (n_0, k_0) with $v < \infty$
$k = mk_0 + k_0$, $n = (m+1)\, n_0$ is called an (n, k) trellis code.

b) Time Invariance – Shifting i/p sequence by lk_0 frames $\Rightarrow$ o/p sequence is shifted by ln_0 frames.

c) Linearity

$$\text{Inputs} \quad \left\{ \begin{array}{l} u_1 \rightarrow c_1 \\ u_2 \rightarrow c_2 \end{array} \right\} \quad \text{outputs}$$

$$\Rightarrow \quad au_1 + bu_2 \rightarrow ac_1 + bc_2$$

$\in$ GF(q)

A linear, time-invariant finite constraint tree code $[(n_0, k_0)]$ is called an (n, k) convolutional code $n \equiv (m+1)n_0$, $k = (m+1)k_0$

An (n_0, k_0) tree code that is time invariant and has finite wordlength k is called an (n, k) sliding block code.

- The GF(2) and convolutional codes (CC).

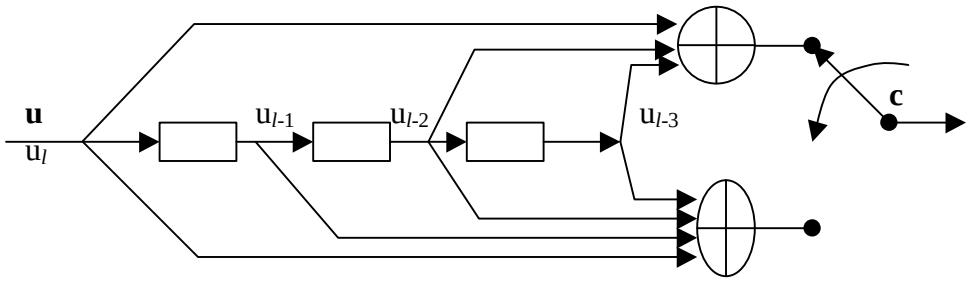
Several representations of a CC

a) Shift register – (Implementation)
b) Tree – (Sequential decoding)
c) Trellis – (Viterbi algorithm)
d) Matrix – (Syndrome decoding)
e) Polynomial – (Catastrophic/noncatastrophic)
f) State diagram – (Distance properties)

Example

$(n_0 = 2, k_0 = 1, m = 3)$ CC
k=4, n=8, R=1/2
$n=(m+1)n_0 \quad k=(m+1)k_0$
n_0 – no. of symbols / output word frame
m – memory frames
k_0 – no. of input symbols/frames

a)



Have 2 (interleaved) output) sequences
up – $\mathbf{c}^{(1)} = (c_0^{(1)}, c_1^{(1)}, c_2^{(1)}, ...$
down – $\mathbf{c}^{(2)} = (c_0^{(2)}, c_1^{(2)}, c_2^{(2)}, ...$
$c = (c_0^{(1)}, c_0^{(2)}, c_1^{(1)}, c_1^{(2)}, c_2^{(1)}, c_2^{(2)}, ...$

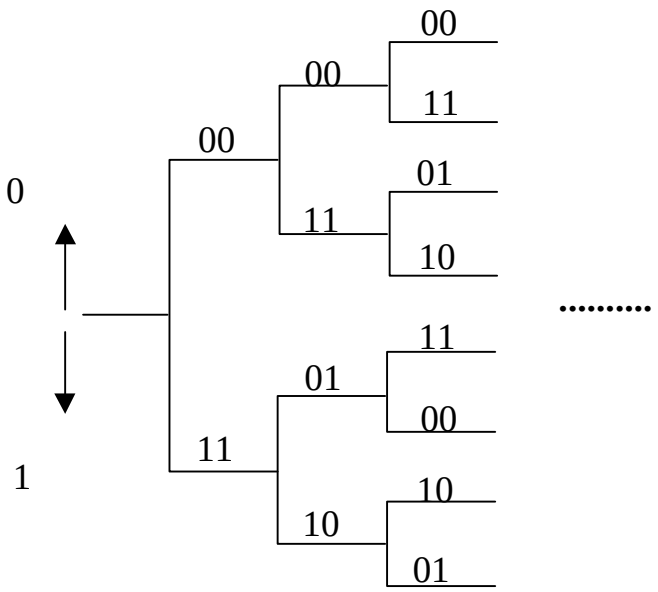
At time t=*l*

GF(2) arithmetic

$$c_l^{(1)} = 1.u_l + 0.u_{l-1} + 1.u_{l-2} + 1.u_{l-3} = \sum_{i=0}^{3} u_{l-i}g_i^{(1)}, \ \mathbf{g}^{(1)} = (1011)$$

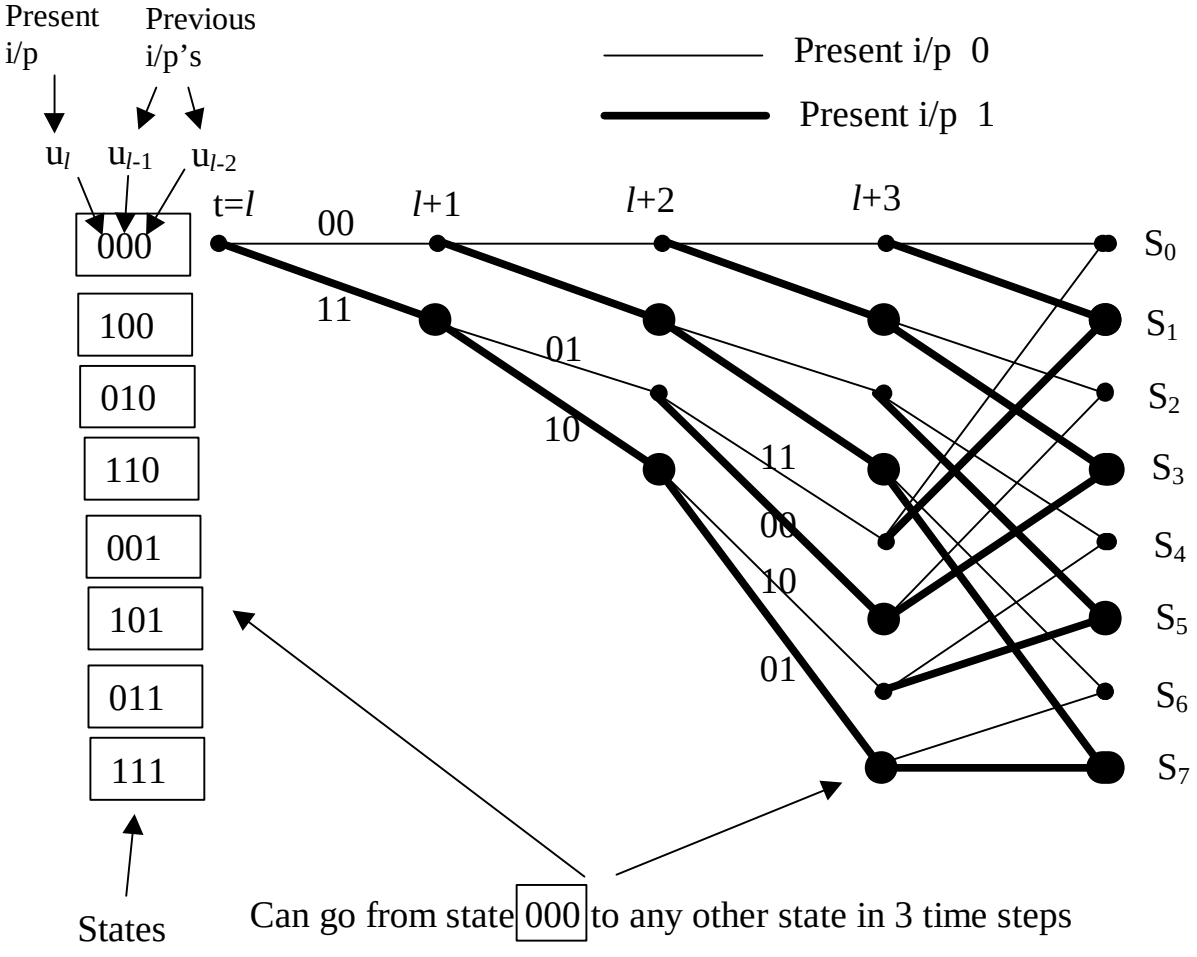
$$c_l^{(2)} = 1.u_l + 1.u_{l-1} + 1.u_{l-2} + 1.u_{l-3} = \sum_{i=0}^{3} u_{l-i}g_i^{(2)}, \ \mathbf{g}^{(2)} = (1111)$$

b) As a tree



c) <u>As a trellis</u>

* Output depends on present input and 3 previous inputs. Define state, s_i, as a 3 bit patterns corresponding to the present i/p and previous 2 i/p's. Thus as the next i/p arrives – we go to a new state (1 of 8) and o/p 2 bits.



c) Matrix

$\mathbf{c} = \mathbf{u} \cdot \mathbf{G}$

assume finite for now
- $(u_0, u_1, u_2, u_3, ..., u_i)$

$(c_0^{(1)}, c_0^{(2)}, c_1^{(1)}, c_1^{(2)}, ..., c_i^{(1)}, c_i^{(2)})$

$$\mathbf{c} = [u_0, u_1, u_2, ..., u_i].\begin{bmatrix} 1\ 1 & 0\ 1 & 1\ 1 & 1\ 1 & 0\ 0 & .... & 0 \\ 0\ 0 & 1\ 1 & 0\ 1 & 1\ 1 & 1\ 1 & & \\ 0\ 0 & 0\ 0 & 1\ 1 & 0\ 1 & 1\ 1 & & \\ 0\ 0 & 0\ 0 & 0\ 0 & 1\ 1 & 0\ 1 & & \\ 0\ 0 & 0\ 0 & 0\ 0 & 0\ 0 & 1\ 1 & & \\ : & & & & & & \end{bmatrix}$$

Rows are as follows [(i+1) rows by 2(i+1) columns.]

$\mathbf{g}^{(1)} = (1011) = (g_0^{(1)}, g_1^{(1)}, g_2^{(1)}, g_3^{(1)})$
$\mathbf{g}^{(2)} = (1111) = (g_0^{(2)}, g_1^{(2)}, g_2^{(2)}, g_3^{(2)})$

Rows are composed of $\mathbf{g}^{(1)}$, $\mathbf{g}^{(2)}$ interleaved $g_0^{(1)}g_0^{(2)}g_1^{(1)}g_1^{(2)}g_2^{(1)}g_2^{(2)}g_3^{(1)}g_3^{(2)}$ and then shifted in steps of 2 to obtain the appropriate output, If output sequence is semi-infinite then **G** is semi-infinite matrix.

(*) Viewed in this manner CC code is a block code!!!

(e) <u>Polynomial Representation</u>

⇒ Output is a convolution of input and coefficients of $\mathbf{g}^{(i)}$

⇒ Transform and output is now a product of input (transformed) and $\mathbf{g}^{(i)}$ (transformed).

⇒ Since discrete, appropriate transform is analogous to Z-transform.

Let $\mathbf{u}(x) = u_0+u_1x+u_2x^2+u_3x^3+ u_4x^4+.....$

power indicates bit position (time delay)

$\mathbf{c}^{(1)}(x) = c_0^{(1)}+ c_1^{(1)}x+ c_2^{(1)}x^2+ c_3^{(1)}x^3+ c_4^{(1)}x^4+.....$
$\mathbf{c}^{(2)}(x) = c_0^{(2)}+ c_1^{(2)}x+ c_2^{(2)}x^2+ c_3^{(2)}x^3+ c_4^{(2)}x^4+.....$

$\mathbf{g}^{(1)}(x) = 1+ 0.x+ x^2+ x^3= 1+ x^2+ x^3$
$\mathbf{g}^{(2)}(x) = 1+ x+ x^2+ x^3$

So, $\mathbf{c}^{(1)}(x) = \mathbf{u}(x)\,\mathbf{g}^{(1)}(x)$ $\quad$ $\mathbf{c}^{(2)}(x) = \mathbf{u}(x)\,\mathbf{g}^{(2)}(x)$

$\mathbf{c}(x) = \mathbf{u}(x)\,[\mathbf{g}^{(1)}(x) + x\,\mathbf{g}^{(2)}(x^2)]$

Or in matrix form,

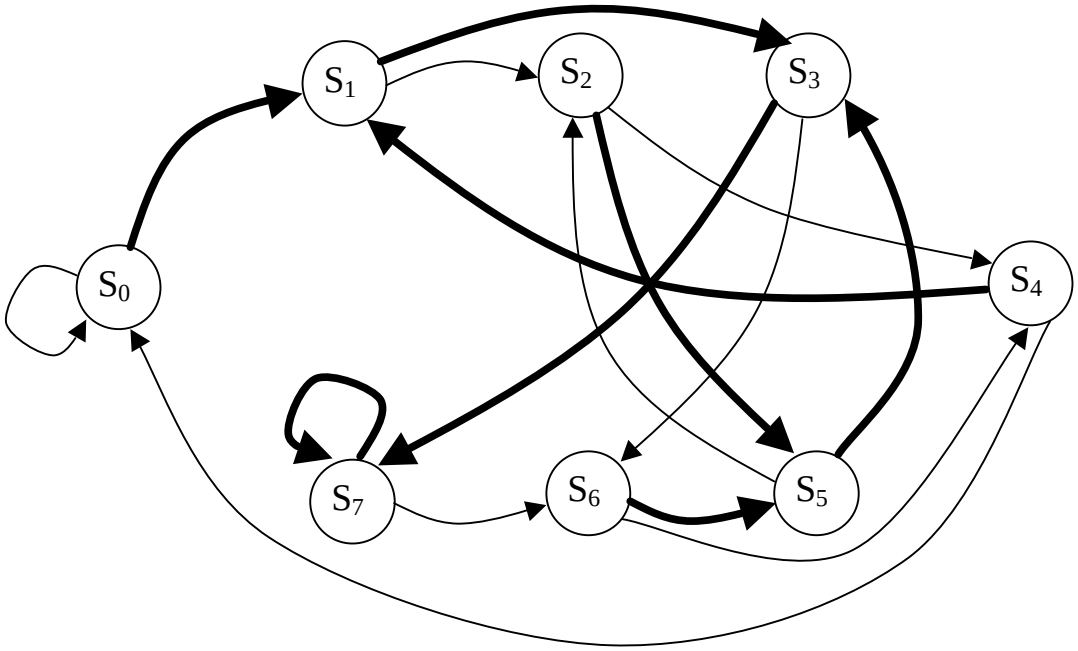
$\mathbf{c}(x) = [\mathbf{c}^{(1)}(x), \mathbf{c}^{(2)}(x)]$ ← Need to multiplex these to get o/p

$= \mathbf{u}(x)\,[\mathbf{g}^{(1)}(x), \mathbf{g}^{(2)}(x)]$ → $G(x)$

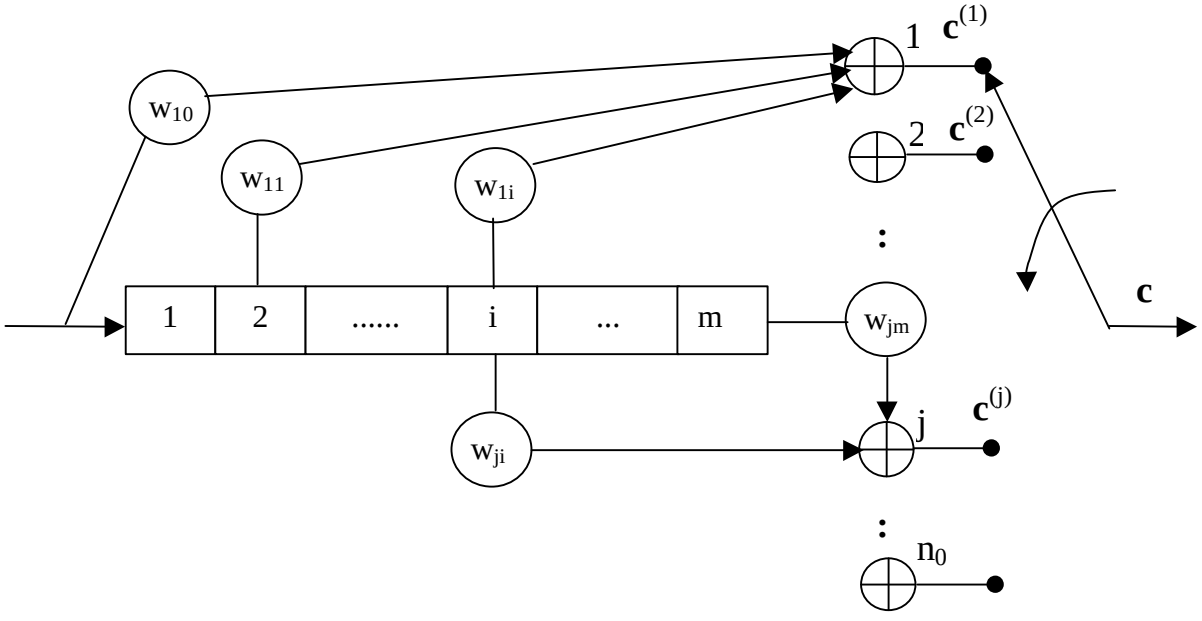
$= \mathbf{u}(x)\,[\,1+x^2+x^3,\ 1+x+x^2+x^3]$

(*) Rather than x one could use notation D(delay) (or z^{-1}).

(e) State Diagram – Have 8 states $S_0, S_1, S_2, \ldots, S_7$



More generally for a $k_0 = 1$, n_0, m CC



Have n_0 interleaved o/p's $\mathbf{c}^{(1)}, c^{(2)}, ..., \mathbf{c}^{(j)}, ..., \mathbf{c}^{(n_0)}$

$$c_l^{(j)} = \mathbf{u} * \mathbf{g}^{(j)} = \sum_{i=0}^{m} u_{l-i} g_i^{(j)} = 0 \text{ or } 1$$

$$\mathbf{g}^{(j)} = (w_{j0}, w_{j1}, w_{j2}, ..., w_{jm})$$

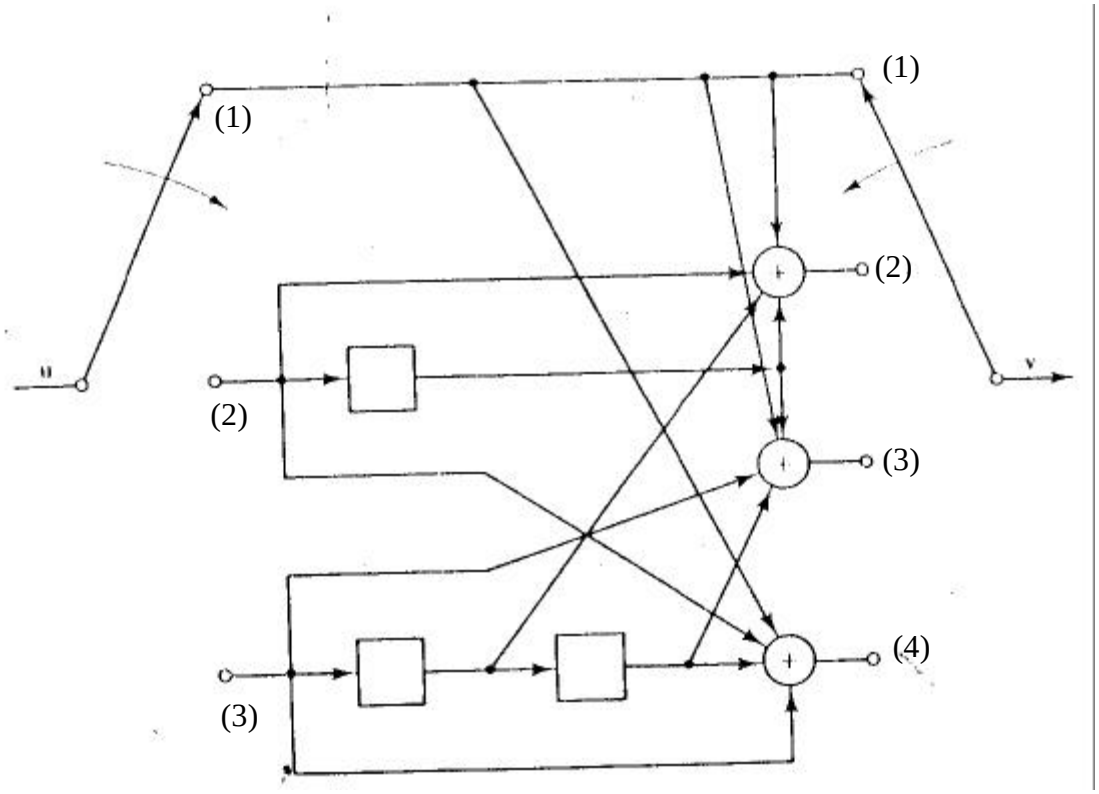
No. of states = $2^n = 2^{k_0 m} = 2^v$ ← constraint length

2 branches leave each state and 2 branches enter each state 2^{k_0}

$$\mathbf{G}(x) = (\mathbf{g}^{(1)}(x), \mathbf{g}^{(2)}(x), ..., \mathbf{g}^{(n_0)}(x))$$

If $k_0 > 1$ – same description hold but extended.

Example $k_0 = 3$, $n_0 = 4$, $m = 2$



Now $n = (m+1)\, n_0 = 12$ $\quad v = mk_0 = 6$

$k = (m+1)\, k_0 = 9$

Again have a tree, trellis, generator matrix, polynomial state diagram representation.

Let $\mathbf{g}_i^{(j)} = \mathbf{g}_{ij}$ $\quad i = 1, 2, ..., k_0$ (i/p)
$\quad j = 1, 2, ..., n_0$ (o/p)

(*) represent the taps from each shift register i to appropriate o/p j.

Here we have
$3 \times 4 = 12$ $\quad \mathbf{g}_{ij}$'s

$\mathbf{g}_{11} = 1\ 0\ 0$ $\quad \mathbf{g}_{21} = 0\ 0\ 0$ $\quad \mathbf{g}_{31} = 0\ 0\ 0$
$\mathbf{g}_{12} = 1\ 0\ 0$ $\quad \mathbf{g}_{22} = 1\ 1\ 0$ $\quad \mathbf{g}_{32} = 0\ 1\ 0$
$\mathbf{g}_{13} = 1\ 0\ 0$ $\quad \mathbf{g}_{23} = 0\ 1\ 0$ $\quad \mathbf{g}_{33} = 1\ 0\ 1$
$\mathbf{g}_{14} = 1\ 0\ 0$ $\quad \mathbf{g}_{24} = 1\ 0\ 0$ $\quad \mathbf{g}_{34} = 1\ 0\ 1$

<u>As a convolution</u>

$$\mathbf{c}^{(1)} = \mathbf{u}^{(1)} * \mathbf{g}_{11} + \mathbf{u}^{(2)} * \mathbf{g}_{21} + \mathbf{u}^{(3)} * \mathbf{g}_{31}$$
$$\mathbf{c}^{(2)} = \mathbf{u}^{(1)} * \mathbf{g}_{12} + \mathbf{u}^{(2)} * \mathbf{g}_{22} + \mathbf{u}^{(3)} * \mathbf{g}_{32}$$
$$\mathbf{c}^{(3)} = \mathbf{u}^{(1)} * \mathbf{g}_{13} + \mathbf{u}^{(2)} * \mathbf{g}_{23} + \mathbf{u}^{(3)} * \mathbf{g}_{33}$$
$$\mathbf{c}^{(4)} = \mathbf{u}^{(1)} * \mathbf{g}_{14} + \mathbf{u}^{(2)} * \mathbf{g}_{24} + \mathbf{u}^{(3)} * \mathbf{g}_{34}$$

Polynomial generator matrix-express each $\mathbf{g}_{ij}$ as a polynomial.

$$\mathbf{G}(x) = \begin{bmatrix} \mathbf{g}_{11}(x) & \mathbf{g}_{12}(x) & \mathbf{g}_{13}(x) & \mathbf{g}_{14}(x) \\ \mathbf{g}_{21}(x) & \mathbf{g}_{22}(x) & \mathbf{g}_{23}(x) & \mathbf{g}_{24}(x) \\ \mathbf{g}_{31}(x) & \mathbf{g}_{32}(x) & \mathbf{g}_{33}(x) & \mathbf{g}_{34}(x) \end{bmatrix}$$

$$\mathbf{c}(x) = [\mathbf{c}^{(1)}(x), \mathbf{c}^{(2)}(x), \mathbf{c}^{(3)}(x), \mathbf{c}^{(4)}(x)]$$
$$= [\mathbf{u}^{(1)}(x), \mathbf{u}^{(2)}(x), \mathbf{u}^{(3)}(x)] \,.\, [\mathbf{G}(x)]$$

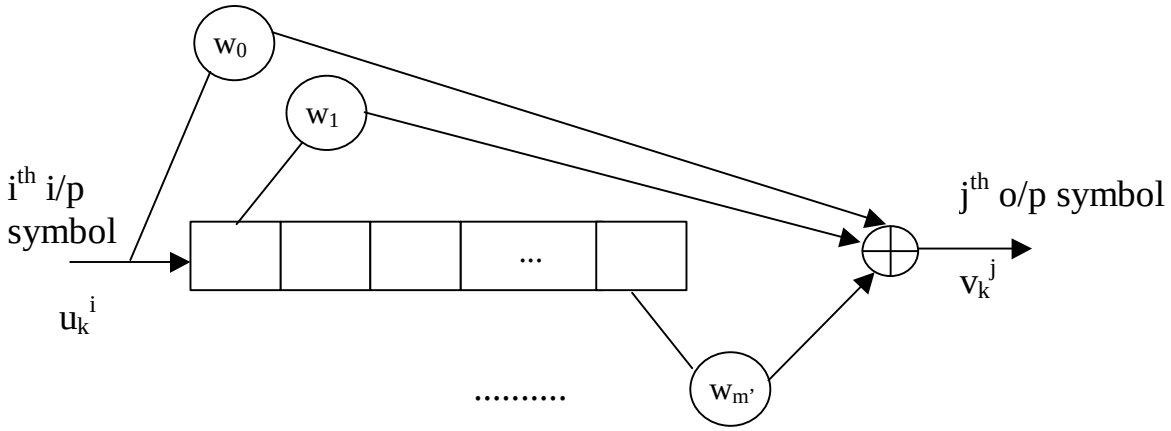
$$\mathbf{G}(x) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1+x & x & 1 \\ 0 & x & 1+x^2 & 1+x^2 \end{bmatrix}$$

Polynomial Description of Convolutional Code

- Have a SR (shift register) from each of the k_0 inputs to each of the n_0 outputs. Each SR can be thought of as a finite response filter (FIR) with a transfer function corresponding.

$\sum_{l=0}^{m'} w_l x^l$ $\quad w_l$ – weight – depends on GF 0, 1 if GF(2)

x – Delay operator – power of x which is *l* denotes delay.
m' – length of SR - $w_{m'} \neq 0$



$$v_k^{(j)} = w_0^{(ij)} u_k^{(i)} + w_1^{(ij)} u_{k-1}^{(i)} + w_2^{(ij)} u_{k-2}^{(i)} + \ldots + w_{m'}^{(ij)} u_{k-m'}^{(i)}$$
$$= \sum_{l=0}^{m'} w_l^{(ij)} u_{k-l}^{\ i}$$

k- corresponds to the instant we are calculating o/p.

Or taking transform

$$\mathbf{v}^{(j)}(x) = \mathbf{u}^{(i)}(x) [w_0^{(ij)} + w_1^{(ij)} x + w_2^{(ij)} x^2 + \ldots + w_{m'}^{(ij)} x^{m'}]$$

call $\mathbf{g}_{ij}(x)$

Define

(a) Constraint length $v \equiv \sum_{i=1}^{k_0} \max_j [\deg \mathbf{g}_{ij}(x)]$

Smaller the constraint length $\Rightarrow$ smaller the trellis
(in terms of states = q^v)
If each SR is of length m (with a nonzero tap at the last stage) then $v=k_0 m$

No. of states = q^{mk_0} (2^{mk_0})

(b) Wordlength – $k \equiv k_0 \,.\, \max_{i,j}[\deg \mathbf{g}_{ij}(x) + 1]$

(c) Blocklength – $n \equiv n_0 \,.\, \max_{i,j}[\deg \mathbf{g}_{ij}(x) + 1]$

Consider now an n_0, k_0, CC

Represent i/p information stream as k_0 information streams represented in the transform domain as a polynomial in the delay operator x, $d_i(x)$ – Have k_0 polynomials where i=1, ..., n_0

Similarly each o/p codeword stream is $c_j(x)$ where j=1, ..., n_0

Now, $\mathbf{c}_j(x) = \mathbf{d}_1(x)\,\mathbf{g}_{1j}(x) + \mathbf{d}_2(x)\,\mathbf{g}_{2j}(x) + ... + \mathbf{d}_{k_0}(x).\mathbf{g}_{k_0 j}(x)$

or $\mathbf{c}(x) = [\mathbf{c}_1(x), ..., \mathbf{c}_{n_0}(x)] = \mathbf{d}(x)\,.\,\mathbf{G}(x)$

$$= [\mathbf{d}_1(x), ..., \mathbf{d}_{k_0}(x)] \begin{bmatrix} \mathbf{g}_{11}(x) & .... & \mathbf{g}_{1,n_0}(x) \\ \mathbf{g}_{21}(x) & .... & \mathbf{g}_{2,n_0}(x) \\ \vdots & & \\ \mathbf{g}_{k_0,1}(x) & .... & \mathbf{g}_{k_0,n_0}(x) \end{bmatrix}$$

Polynomial Generator Matrix

k_0 x n_0

Parity-check polynomial matrix, $\mathbf{H}(x)$ is one such that

$$\mathbf{G}(x).\mathbf{H}^T(x) = 0$$

k_0 x n_0 $\qquad$ n_0 x (n_0 - k_0)

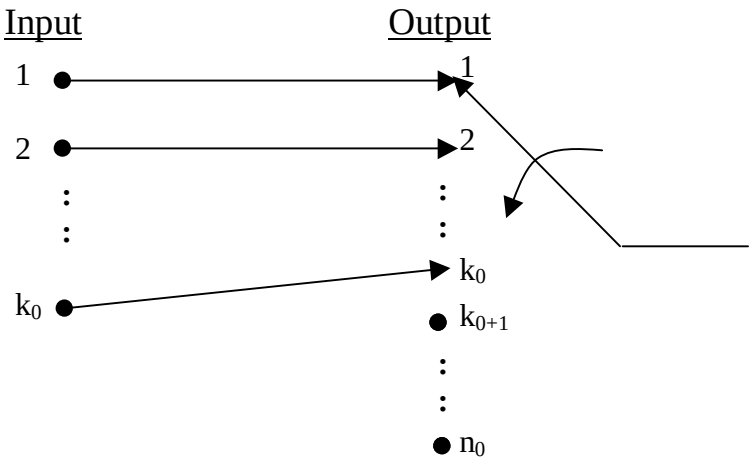
Syndrome polynomial vector $\Rightarrow \mathbf{S}(x) = \mathbf{R}(x).\,\mathbf{H}^T(x)$

Systematic Encoder One in which the 1st k_0 symbols of the n_0 symbols in the o/p frame are identitical to the o/p frame.

i.e. $\mathbf{g}_{jj}(x) = 1$, $\mathbf{g}_{ij}(x) = 1$ $\quad j = 1, 2, ..., k_0$; $i = 1, 2, ..., k_0$; $i \neq j$

$\mathbf{g}_{11}(x) = 1$, $\mathbf{g}_{i1}(x) = 0$ $\quad i = 2, 3, 4, ..., k_0$

$\mathbf{g}_{22}(x) = 1$, $\mathbf{g}_{i2}(x) = 0$ $\quad i = 1, 3, 4, ..., k_0$



Thus

$\mathbf{G}(x) = [\ \mathbf{I}\ |\ \mathbf{P}(x)]$ $\qquad$ $\mathbf{H}(x) = [-\mathbf{P}^T(x)\ |\ \mathbf{I}]$

$k_0 \times n_0$ $\qquad$ $n_0 \times (n_0 - k_0)$ $\qquad$ $(n_0 - k_0) \times (n_0 - k_0)$

Catastrophic Codes

- one where an i/p sequence of infinite (Hamming) weight produces an o/p sequence of finite weight.

To recover the information sequence from the codeword

$\mathbf{c}(x) = \mathbf{d}(x)\ .\ \mathbf{G}(x)$

Find $\mathbf{G}^{-1}(x)$ such that $\mathbf{G}(x).\mathbf{G}^{-1}(x) = \mathbf{I}\ x^l$

$k_0 \times n_0$ $\quad$ $n_0 \times k_0$ $\quad$ $k_0 \times k_0$

Then $\mathbf{c}(x).\mathbf{G}^{-1}(x) = \mathbf{d}(x).\ \mathbf{G}(x).\mathbf{G}^{-1}(x) = d(x)\ .\ \mathbf{I}\ x^l$

$= \mathbf{d}(x)\ x^l$

delayed version of i/p

(Procedures to construct $\mathbf{G}^{-1}(x)$ given in

(1) Forney – IEEE, Information Theory, IT-16, No. 6, Nov. 1970, pp. 720-738
"Convolutional Codes I : Algebraic Structure"

(2) Massey/Sain – IEEE Trans. COmputers – C – 17
pp. 330-337, April 1968)

Basically if $\mathbf{G}^{-1}(x)$ exits then code is noncatastrophic (necessay and sufficient condition)

or for $k_0 = 1$ $GCD[\mathbf{g}_{11}(x), \mathbf{g}_{12}(x),..., \mathbf{g}_{1,n_0}(x)] = x^l$

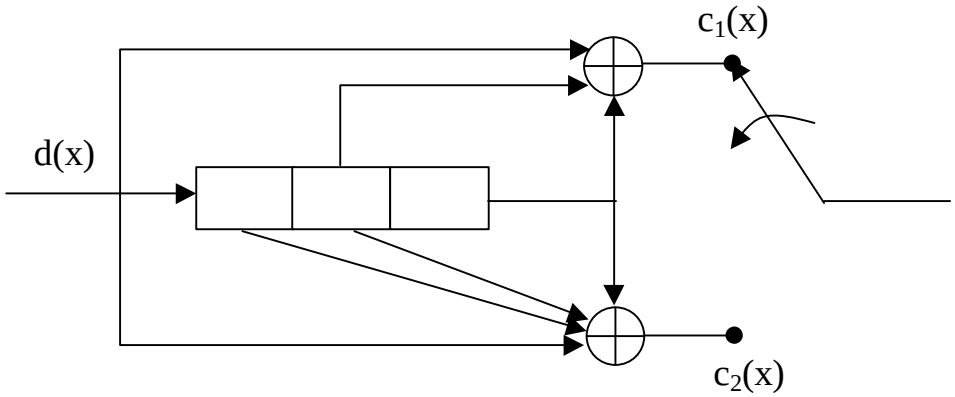
for k_0 general

$$GCD[\Delta_i(x), i=1, 2, ..., \binom{n_0}{k_0}] = x^l$$

$\Delta_i(x)$ – Determinants of the $\binom{n_0}{k_0}$ distinct submatrices of $\mathbf{G}(x)$.

Example

$k_0 = 1$, $n_0 = 2$, $m = 3$



$\mathbf{g}_{11}(x) = 1 + x^2 + x^3$ $\mathbf{g}_{12}(x) = 1 + x + x^2 + x^3$

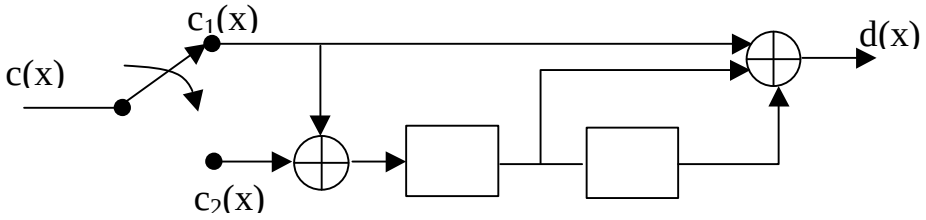
$\mathbf{G}(x) = [1 + x^2 + x^3, 1 + x + x^2 + x^3]$

$GCD\,[1 + x^2 + x^3, 1 + x + x^2 + x^3] = 1$

$\Rightarrow$ $\mathbf{G}^{-1}(x)$ exists $\Rightarrow$ code is noncatastrophic

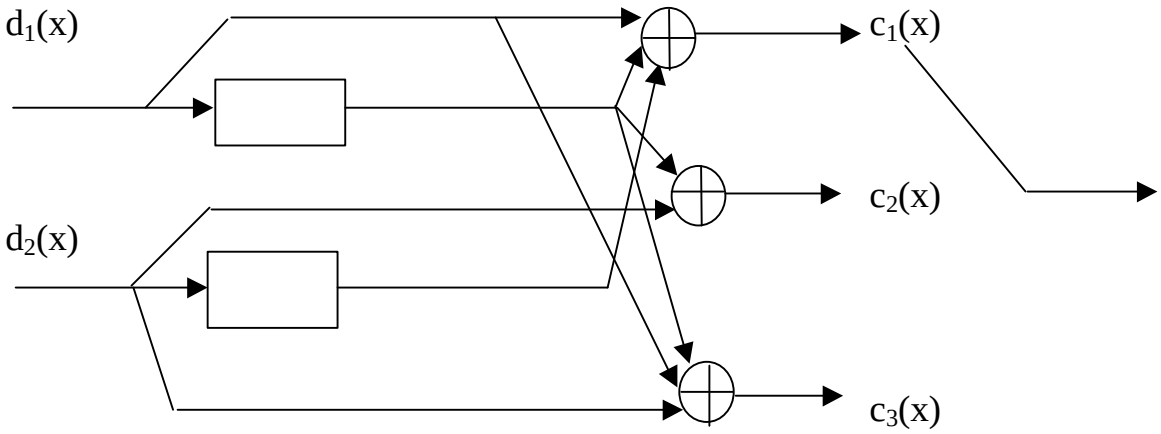
$$\mathbf{G}^{-1}(x) = \begin{bmatrix} 1 + x + x^2 \\ x + x^2 \end{bmatrix}$$ verify ???

Inverse looks as follows



Example

$k_0 = 2, n_0 = 3$



$$\mathbf{G}(x) = \begin{bmatrix} 1+x & x & 1+x \\ x & 1 & 1 \end{bmatrix}$$

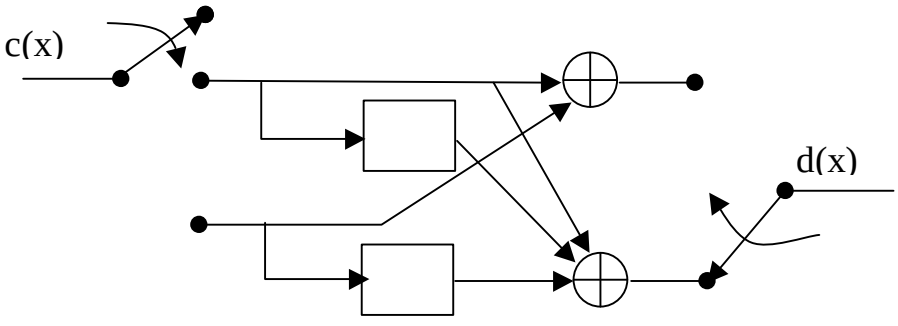
2 x 2 submatrix – have determinants of

$1+x+x^2,\ 1,\ 1+x^2$

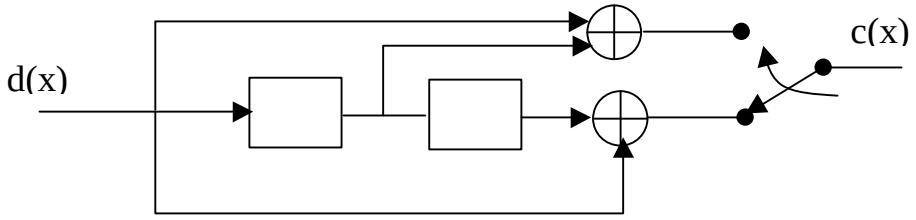
GCD $[1+x+x^2,\ 1,\ 1+x^2] = 1$

$$\mathbf{G}^{-1}(x) = \begin{bmatrix} 0 & 0 \\ 1 & 1+x \\ 1 & x \end{bmatrix}$$

Inverse



Example



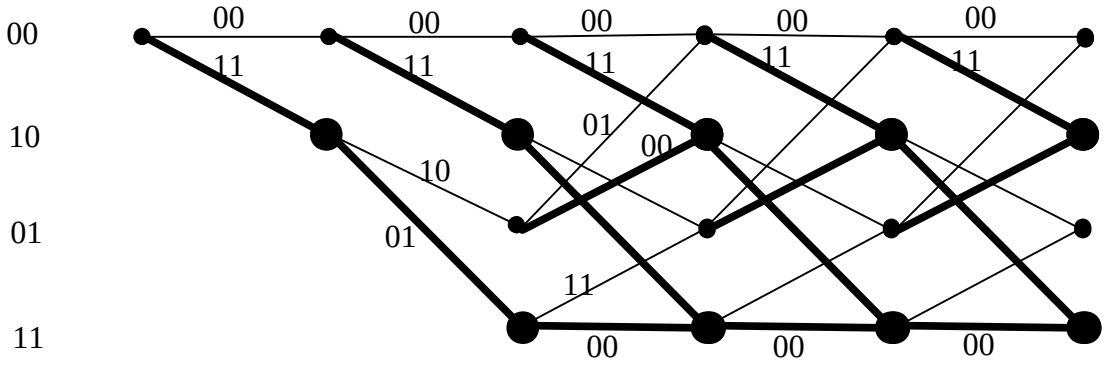
$\mathbf{G}(x) = [1+x, 1+x^2]$

$GCD[1+x, 1+x^2] = 1+x \neq x^l$

Þ $\mathbf{G}^{-1}(x)$ dose not exist.

⇒ catastrophic code

Consider - the trellis



Consider input sequence

1 1 1 1 ....

$$= 1 + x + x^2 + x^3 + x^4 + \ldots = \frac{1}{1+x}$$

The o/p sequence is 11 01 00 00 00 ...

or	$\mathbf{c}_1$	1↓	0↓	0↓	0↓	0↓	= 1
	$\mathbf{c}_2$	1	1	0	0	0	= (1 + x)

Note that 3 errors on the decoded o/p result in a legitimate codeword, i.e, 0 ⇒ infinite number of decoded errors ⇒ catastrophic.

- In terms of state diagram – code is catastrophic if and only if state diagram contains a loop (other than self-loop of state 0) of zero weight.
- Systematic code – always noncatastrophic.

- For (n_0, k_0=1) codes only $\frac{1}{2^{n_0}-1}$ codes are catastrophic.

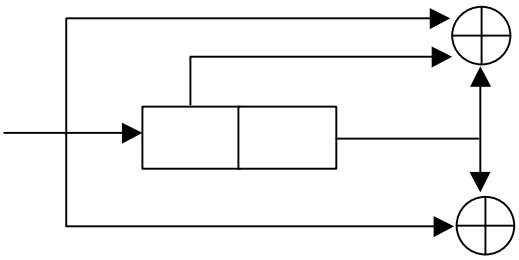
Distance Properties of Convolutional Code

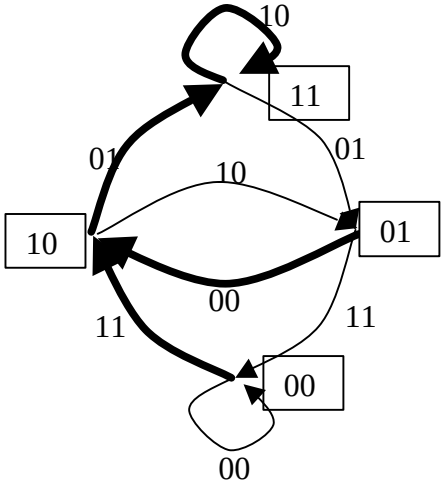
- Coding can be viewed as a linear block code

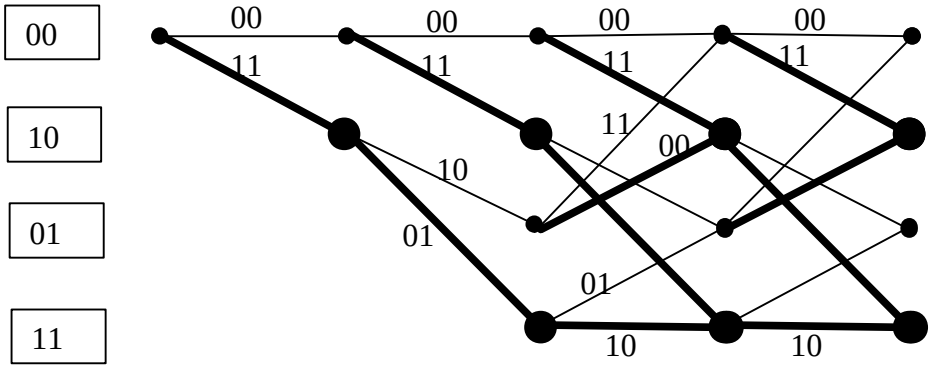
⇒ find distance of all codewords from zero codeword.

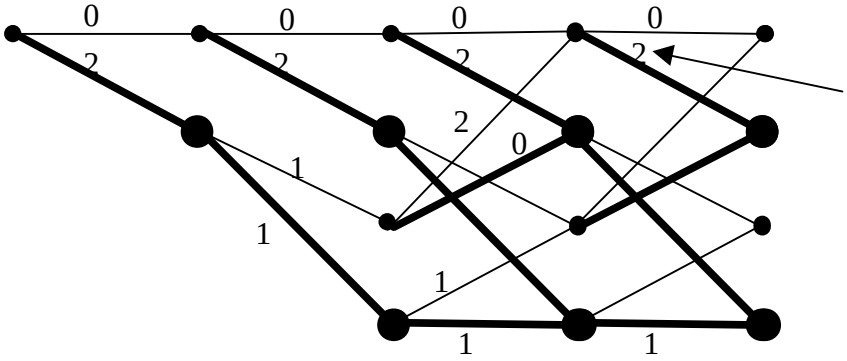
- can do this by inspecting the trellis

Use State Diagram





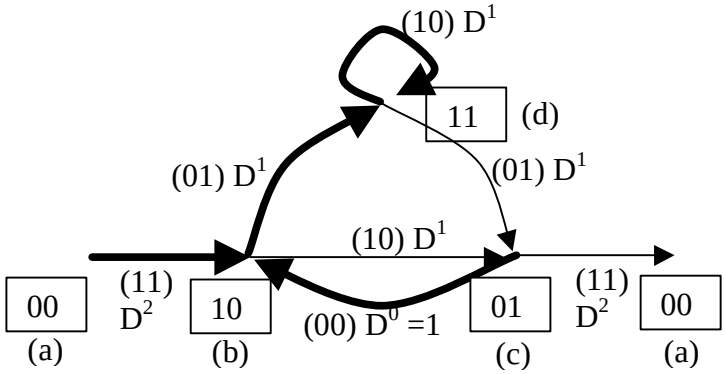




Hamming distance to all zero paths

To find distance from all paths to all zero path, consider all paths that start oo, diverge from it and subsequently terminate at state 00.

Redraw the state diagram as follows



Signal flow graph

D – (gain) – power of D – represents distance of a particular branch from the corresponding all zero path.

Define – intermediate variables – E at each state

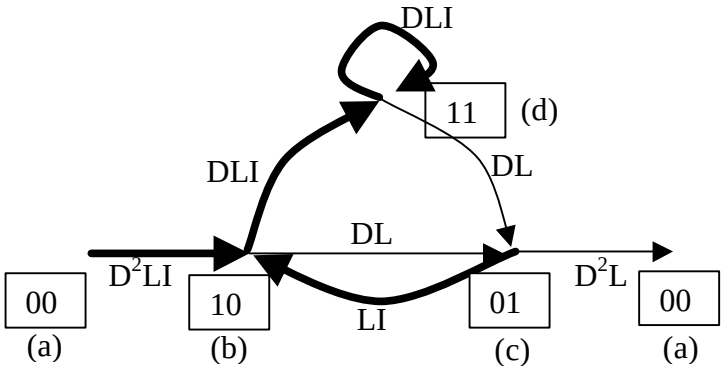
$E_b = D^2 + E_c$ o/p (a) $= D^2 E_c = T(D)$
$E_c = D\,E_d + D\,E_b$ Transfer function when i/p
$E_d = D\,E_b + D\,E_d$ is an "impulse"

$$\begin{bmatrix} 1 & -1 & 0 \\ -D & 1 & -D \\ -D & 0 & 1-D \end{bmatrix} \begin{bmatrix} E_b \\ E_c \\ E_d \end{bmatrix} = \begin{bmatrix} D^2 \\ 0 \\ 0 \end{bmatrix} \qquad E_c = \frac{D^3}{1-2D}$$

$$T(D) = \frac{D^5}{1-2D} = D^5 + 2D^6 + 4D^7 + \ldots + 2^k D^{k+5} + \ldots$$

1 path at distance 5; 2 paths at distance 6; 2^k paths at distance k+5 — paths that deviate from & then merge back with all zero path

Considerably more informaion can be obtained by this approach.
Let L – variable – representing path length.
I - variable – representing input sequence weight



$$T(D,\ L,\ I) = \frac{D^5L^3I}{1 - DL(L+1)I} = D^5L^3I + D^6L^4(1+L)I^2 + D^7L^5(1+L)^2I^3 + \ldots + D^{(k+5)}L^{(k+3)}\,(L+1)^kI^{(k+1)} + \ldots$$

Have 1 path – distance 5 from all zero codeword
- length 3
- input sequence has on 1

Have 2 paths – distance 6
- one of length 4, other length 5
- input sequence has two 1's

or Have 2 paths of length 6 – $(L^5(1+L)^2 = L^5+2L^6+L^7)$

Note

$$T(D, L, I)|_{L=1,\ I=1} = T(D) = \sum_{d=d_f}^{\infty} a(d)\, D^d$$

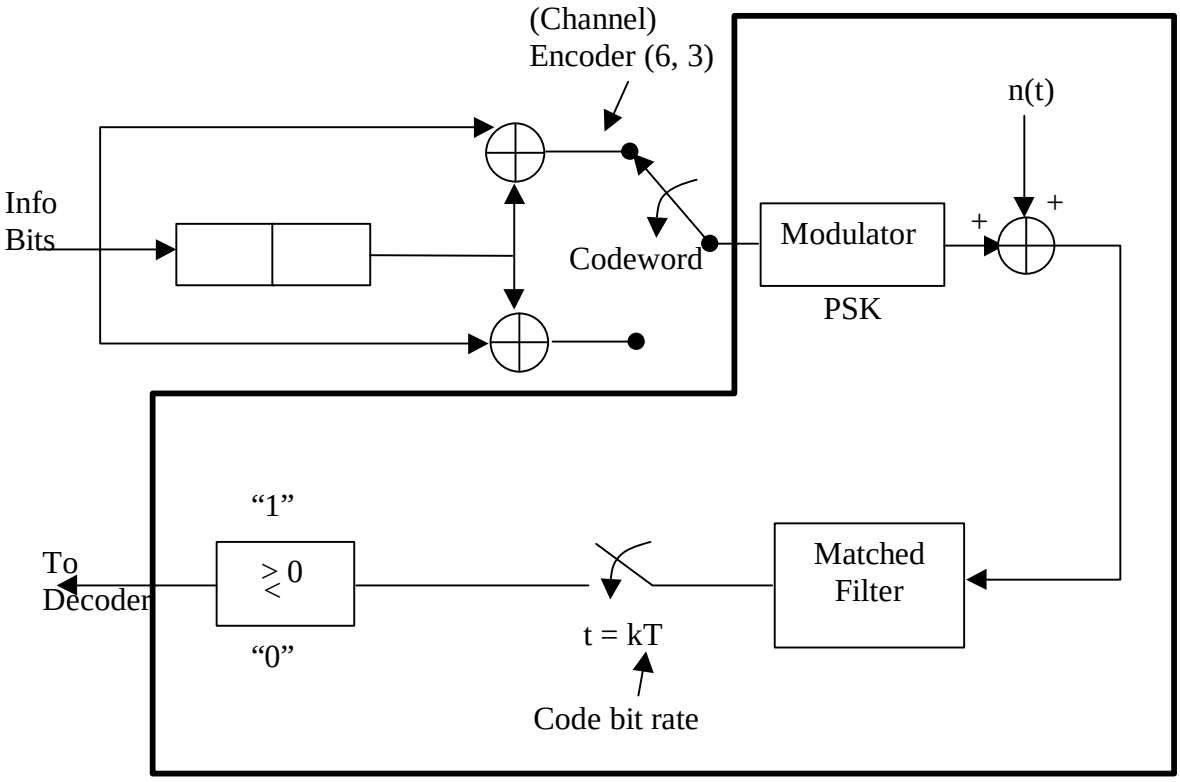
d_f: free distance

$a(d)$: no. of code paths that merge at a given step of weight d

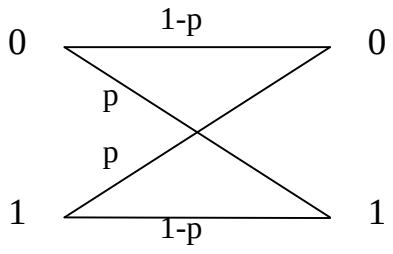
Decoding of Convolutional Codes

Viterbi Decoder
- Maximum Likelihood Sequence Decoder (MLSD, MLSE)
- Complete
- Practical for constraint lengths up to 7-10 ⇒ No. of states 2^7 - 2^{10}

Illustrate by an example

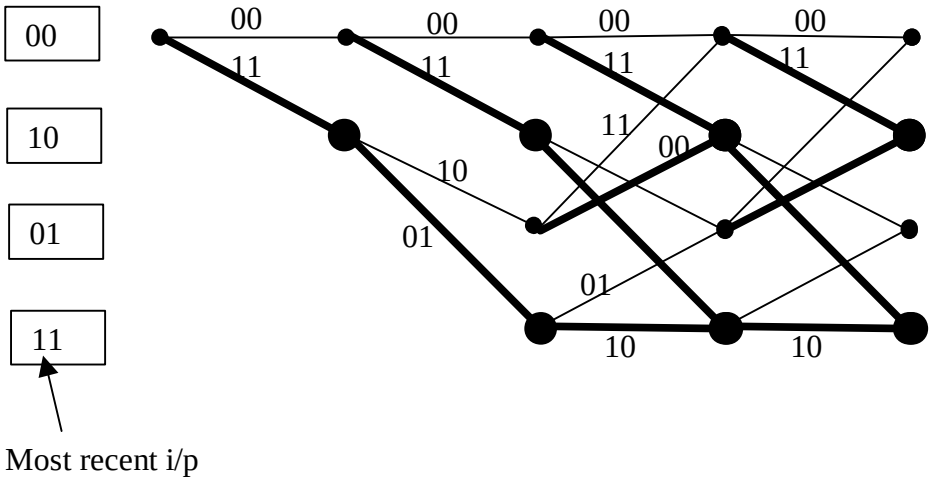


BSC (DMC)



MLD ⇒ Minimum Hamming Distance

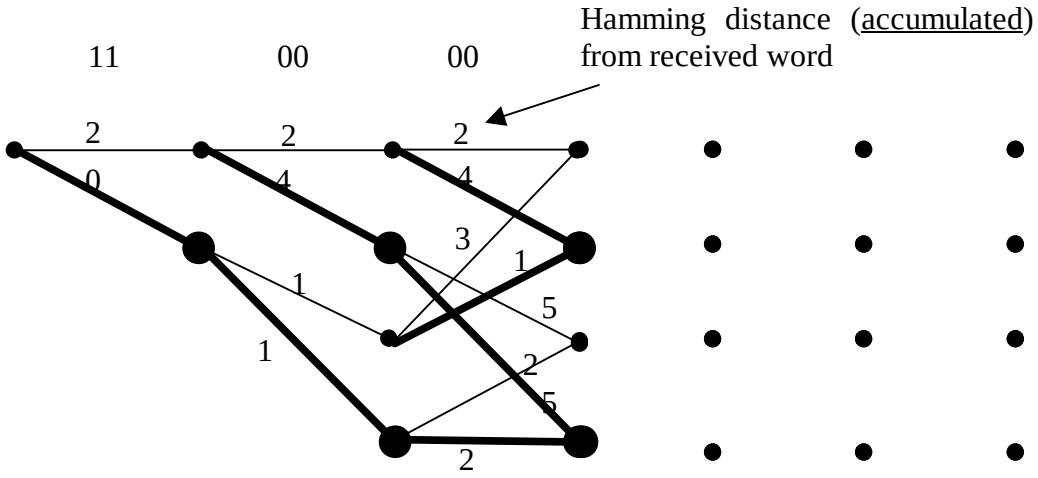
Trellis Representation



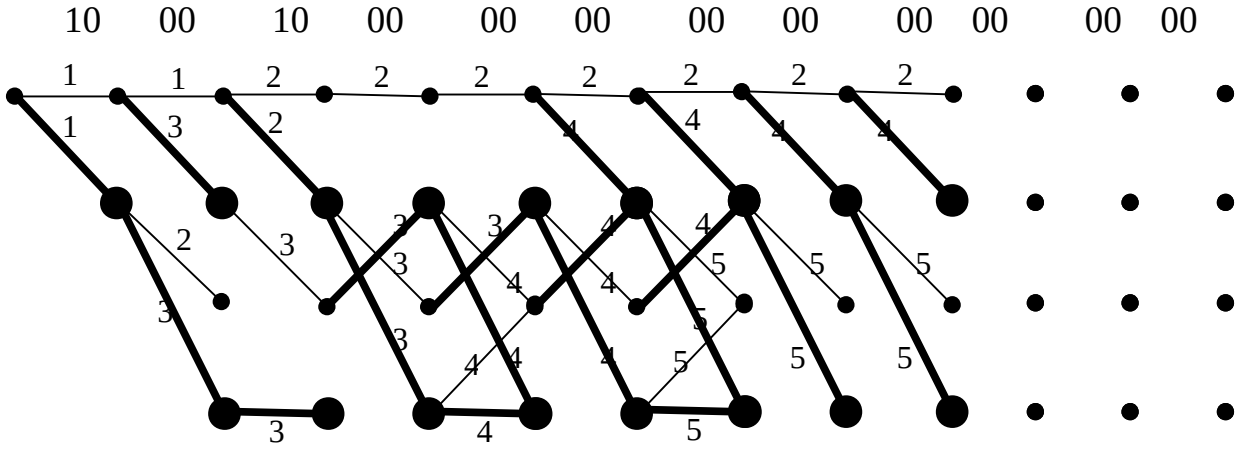
Assume that the output of the BSC (DMC) is

T = 11 00 00 00 00 ....., i.e., all zero codeword with 2 errors.

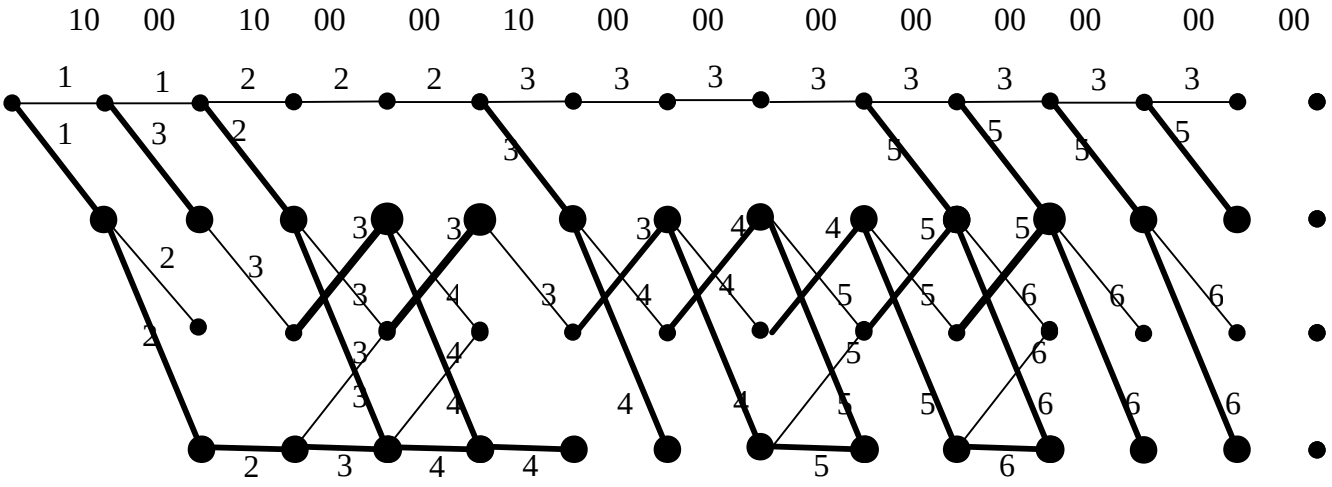
- wish to find codeword closest to received word in a Hamming distance sense.



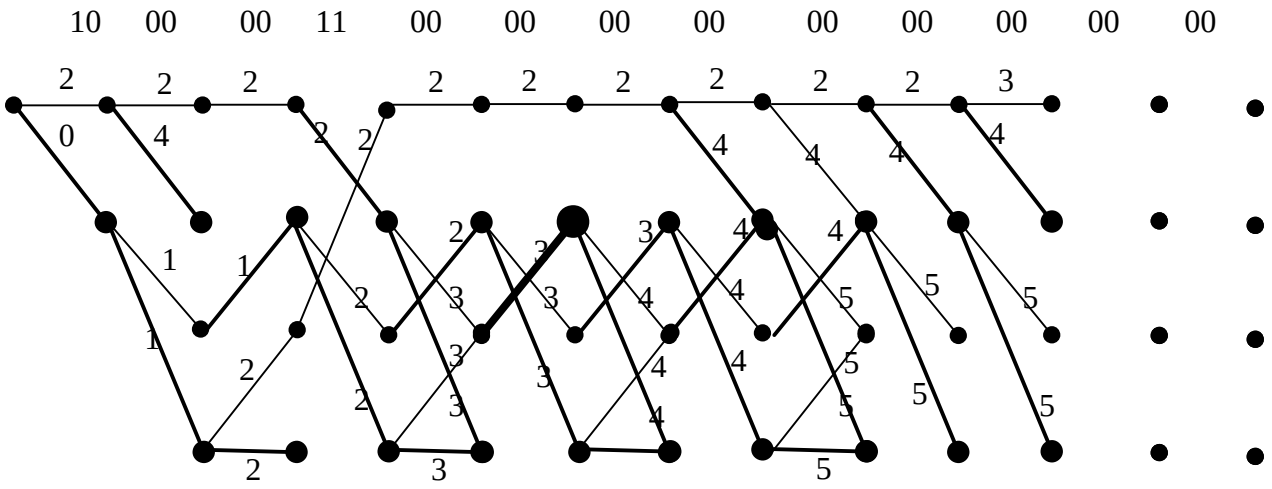
Another example



Still another example



Finally,



Rule of thumb – merges occur with $\underline{\sim}$ 5 constraint lengths
above v = 2 $\Rightarrow$ merge occurs within $\underline{\sim}$ 10 steps.

<u>More generally</u>

$N = n_0 d \Rightarrow$ d=depth of trellis

MLD chooses $\hat{\mathbf{c}}$ which maximizes

$$\Pr[\hat{\mathbf{R}} \,|\, \hat{\mathbf{c}}] = \prod_{i=1}^{d} \log \Pr[\hat{R}_i \,|\, \hat{c}_i] = \prod_{i=1}^{n_0 d} \Pr[R_i \,|\, c_i]$$

$$\log \Pr[\hat{\mathbf{R}}|\hat{\mathbf{c}}] = \sum_{i=1}^{d} \log \Pr[\hat{R}_i|\hat{c}_i] = \sum_{i=1}^{n_0 d} \log \Pr[R_i|c_i]$$

↑ path metric for $\hat{\mathbf{c}}$ ↑ branch metric ↑ bit (symbol) metric

when $r_i \neq c_i$ $\Pr(R_i|c_i) = p$, $r_i = c_i \Rightarrow \Pr(R_i|c_i) = 1 - p$

If $d(\hat{\mathbf{R}}, \hat{\mathbf{c}})$ is the distance between $\hat{\mathbf{R}}$, $\hat{\mathbf{c}}$

$$\log\{ \Pr[\hat{\mathbf{R}}|\hat{\mathbf{c}}] \} = d(\hat{\mathbf{R}}, \hat{\mathbf{c}}) \log p + [N - d(\hat{\mathbf{R}}, \hat{\mathbf{c}})] \log(1-p)$$

$$= d(\hat{\mathbf{R}}, \hat{\mathbf{c}}) \log \frac{p}{1-p} + N \log(1-p)$$

Since $\log[\frac{p}{1-p}] < 0$ for $p < ½$, N log (1-p) is a constant for all $\hat{\mathbf{c}}$.

$\Rightarrow$ <u>MLD $\Rightarrow$ Minimize $d(\hat{\mathbf{R}}, \hat{\mathbf{c}})$</u>

$$d(\hat{\mathbf{R}}, \hat{\mathbf{c}}) = \sum_{i=1}^{d} d(\hat{R}_i|\hat{c}_i) = \sum_{i=1}^{n_0 d} d(R_i|c_i)$$

path metric branch metric bit metric

Thus MLD for a BSC chooses the minimum Hamming distance path from the received sequence.

<u>Partial path metric</u> for first j branches

$$M\{(\hat{\mathbf{R}}, \hat{\mathbf{c}})_j\} = \sum_{i=1}^{j} M(\hat{R}_i|\hat{c}_i)$$

<u>Viterbi Algorithm</u>

1) Compute partial metric for each single path entering each state at some time, say k. Store the best path to each state (survivor) and its metric (state metric).
2) Go on step further into the trellis. Determine the branch metric for al branches terminating at a state, add to the state metric, choose largest – store survivor and metric.
   - repeat for all states.
3) Repeat 2. Stop if j = N.

Sequential Decoding

⇒ Only practical method for large constraint length (v), say v > 10.

⇒ VA requires $\underset{\sim}{} 2^v$ computations per step regardless of noise level.

⇒ For low noise levels – might become "obvious" which path is the likely ML choice – pursue the path and its descendents, i.e., establish a decoding scheme that "adapts" to the noise level.

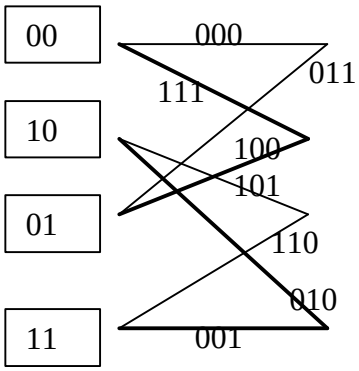
⇒ No. of computations reduced.

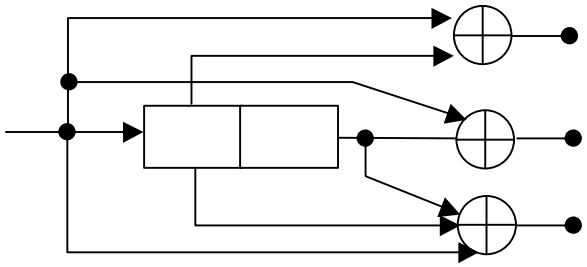
Example

$(n_0 = 3, k_0 = 1)$ code with m=2, and

$\mathbf{G}(x) = [\ 1+x,\ 1+x^2,\ 1+x+x^2\ ]$

Trellis





(9, 6) CC v=2

Codeword transmitted over BSC

⇒ MLD = Hamming distance

$\mathbf{R}$ = (010, 010, 001, 110, 100, 101, 011)

(Received sequence)

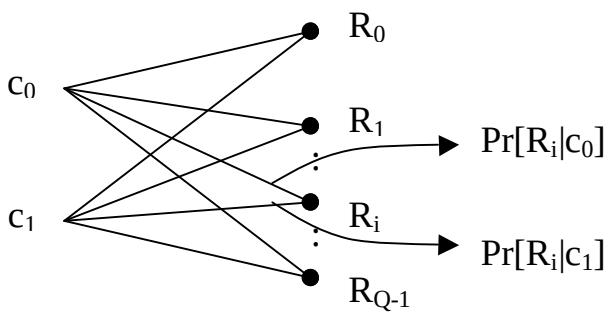
Consider the following 2 paths of different lengths.

			Hamming Distance
1)	$(\hat{\mathbf{c}})_6$ = (111, 010, 001, 110, 100, 101, 011)	⇒	2
2)	$(\hat{\mathbf{c}})_0$ = (000)	⇒	1

Path 2 metric is better than path 1 but intuitively $(\hat{\mathbf{c}})_6$ is a better choice (more likely) to be part of the ML path since $(\hat{\mathbf{c}})_0$ is more likely to accumulate additional distance.

Need to adjust the metric to account for different path lengths.

For a binary i/p Q-ary output DMC, i.e.,



Best metric to use when comparing paths of different lengths is :

$$M(R_i|c_i) = \log_2 \left[\frac{\Pr[R_i \mid c_i]}{\Pr[R_i]} \right] - R$$

(rate)

(1^{st} proposed by Fano on intuitive grounds.)

Partial path metric for 1^{st} l branches of path $\hat{c}$ is :

$$M[(\hat{\mathbf{R}}|\hat{\mathbf{c}})_{l-1}] = \sum_{j=0}^{l-1} M(\hat{R}_j|\hat{c}_j) = \sum_{i=0}^{n_0 l-1} M(R_i|c_i)$$

$$= \sum_{i=0}^{n_0 l-1} \log_2\{\Pr[R_i|c_i]\} - \sum_{i=0}^{n_0 l-1} \log_2\{\Pr[R_i]\} - n_0 l R$$

Metric used in Viterbi's Alg. (n_0 bits per branch)

<u>If channel is symmetric</u>

i.e., $\Pr[j|0] = \Pr[Q-1-j|1]$; $j = 0, 1, ..., Q-1$

$\Rightarrow$ Output symbols are equally likely when i/p symbols are equally likely.

$$M[(\hat{\mathbf{R}}|\hat{\mathbf{c}})_{l-1}] = \sum_{i=0}^{n_0 l-1} \log_2\{\Pr[R_i|c_i]\} + n_0 l\,[\log_2 Q - R]$$

0 for $Q \geq 2$, $R \leq 1$
represents a positive bias that increases linearly with path length

- For a BSC transition probability p, bit metrics are

$M(R_i|c_i) = \log_2 p - \log_2 ½ - R = \log_2 (2p) - R$; $R_i \neq c_i$

$= \log_2 [2(1-p)] - R$; $R_i = c_i$

Example

Pr =1/3 ; p = 0.1

$M(R_i|c_i)$

c_i \ R_i	0	1
0	0.52	-2.65
1	-2.65	0.52

Scale ⇒

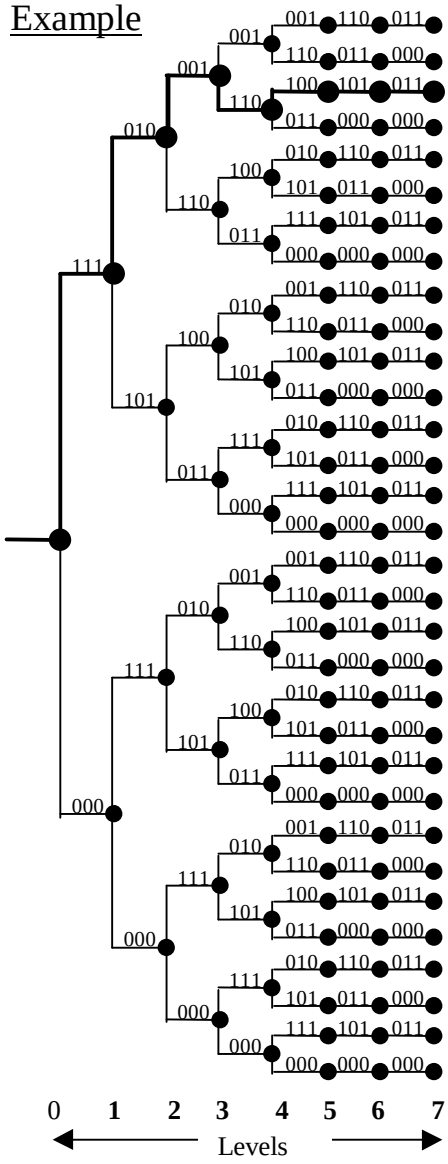
To obtain integer metric table

c_i \ R_i	0	1
0	1	-5
1	-5	1

Stack Algorithm for Sequential Decoding

1) Load stack with origin node of tree, whose metric is taken to be zero.
2) Compute metric of successors of the top path in the stack.
3) Delete top path from the stack.
4) Insert new paths in the stack and rearrange stack in order of decreasing metrics.
5) If not at terminal node, return to 2, otherwise stop.

Example



Code tree for a (3, 1, 2) code with L=5

$\hat{\mathbf{R}}$ = (010, 010, 001, 110, 100, 101, 011)

Use integer metric table

Step 1	Step 2	Step 3	Step 4	Step 5
0(-3)	00(-6)	000(-9)	1(-9)	11(-6)
1(-9)	1(-9)	1(-9)	0001(-12)	0001(-12)
	01(-12)	01(-12)	01(-12)	01(-12)
		001(-15)	001(-15)	001(-15)
			0000(-18)	0000(-18)
				10(-24)

metric

i/p frame

Step 6	Step 7	Step 8	Step 9	Step 10
111(-3)	1110(0)	11101(+3)	111010(+6)	1110100(+9)
0001(-12)	0001(-12)	0001(-12)	0001(-12)	0001(-12)
01(-12)	01(-12)	01(-12)	01(-12)	01(-12)
001(-15)	001(-15)	11100(-15)	11100(-15)	11100(-15)
0000(-18)	1111(-18)	001(-15)	001(-15)	001(-15)
110(-21)	0000(-18)	1111(-18)	1111(-18)	1111(-18)
10(-24)	110(-21)	0000(-18)	0000(-18)	0000(-18)
	10(-24)	110(-21)	110(-21)	110(-21)
		10(-24)	10(-24)	10(-24)

Input bits – end of tree

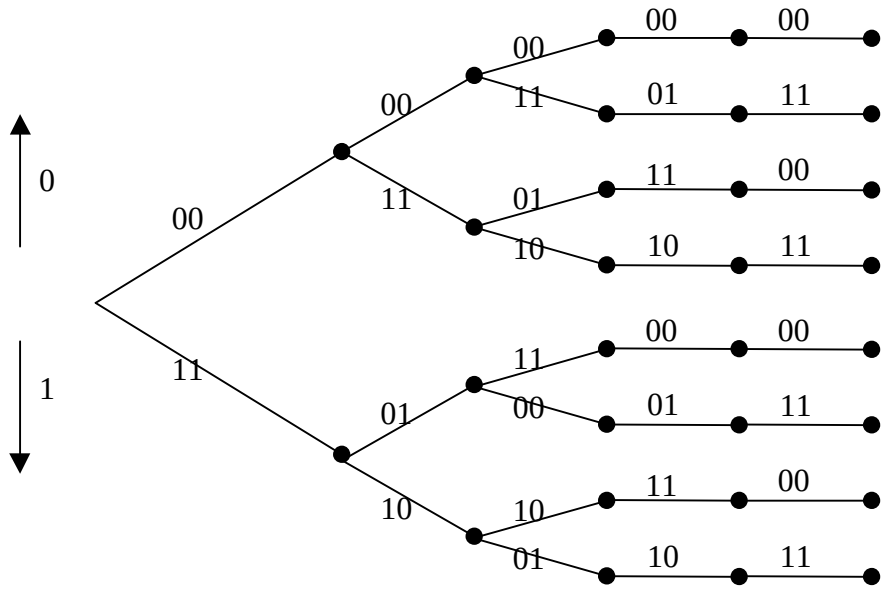
⇒ input **u** = (1110100)

- Ties resolved by placing longest path on top (reduces total no. of decoding steps)
- 10 computations – seq. versus 15 for VA

Derivation of the Fano Metric (n_0, $k_0 = 1$) CC

Thought Experiment

Consider $n_0=2$, $k_0=1$, m=2 CC with following tree.



- Have explored tree to points A, B, C, D
- Next step is to extend the most promising path?
- Which is most promising?

Problem looks like one of decoding a code whose codewords have different lengths.

- So, let's consider this problem.

Let $\{\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_{M-1}\}$ be a code – M different codewords.

$\hat{\mathbf{x}}_i = (x_{i_1}, x_{i_2}, ..., x_{i_{n_i}})$ – length n_i

Let $n \equiv \max n_i$

Transmit code over a DMC:

$$\Pr[\hat{\mathbf{x}}_i \text{ transmitted}] = p_i ; \quad \sum_{i=0}^{M-1} p_i = 1$$

When n_i components of $\hat{x}_i$ have transmitted, add a "random tail" of $n-n_i$ channel i/p symbols. Symbols are selected independently according to a fixed probability distribution p(x) on the channel input alphabet $\{\mathbf{X}\}$.
Receiver always receives an n-symbol sequence

$$\hat{\mathbf{y}} = (y_1, y_2, ..., y_n)$$

To minimize sequence decoding error probability

$$\text{maximize}_i \Pr[\hat{\mathbf{x}}_i \text{ sent} \mid \hat{\mathbf{y}} \text{ received}] = \frac{\Pr[\hat{\mathbf{x}}_i \text{ sent}, \hat{\mathbf{y}} \text{ received}]}{\Pr[\hat{\mathbf{y}}]}$$

$\Pr[\hat{\mathbf{y}}]$ → not a function of i

So, maximize

$$\Pr[\hat{\mathbf{x}}_i \text{ sent}, \hat{\mathbf{y}} \text{ received}] = \Pr[\hat{\mathbf{x}}_i \text{ sent}] \cdot \Pr[\hat{\mathbf{y}} \text{ received} \mid \hat{\mathbf{x}}_i \text{ sent}]$$

$$= \max_i \left\{ p_i \prod_{j=1}^{n_i} \Pr[y_i \mid x_{ij}] \cdot \prod_{j=n_i+1}^{n} \Pr[y_i] \right\}$$

$\prod_{j=1}^{n_i} \Pr[y_i \mid x_{ij}]$: channel transition probabilities

$\prod_{j=n_i+1}^{n} \Pr[y_i]$: probability distribution on channel o/p induced by the probability distribution of random tail

Divide by $\prod_{j=1}^{n} \Pr[y_i]$ - take log and maximize

So,
$$\max_i \left\{ \log p_i + \sum_{j=1}^{n_i} \log \frac{\Pr[y_i \mid x_{ij}]}{\Pr[y_i]} \right\}$$

$$= \max_i \underbrace{\sum_{j=1}^{n_i} \left\{ \log \frac{\Pr[y_i \mid x_{ij}]}{\Pr[y_i]} - \frac{1}{n_i} \log \left(\frac{1}{p_i} \right) \right\}}_{\mu(\hat{\mathbf{x}}_i)}$$

Receiver therefore computes $\mu(\hat{\mathbf{x}}_i)$, i=0, 1, ..., M-1 for a received $\hat{\mathbf{y}}$ and chooses largest.

*** Returning to the trees** – situation is the same as above since without exploring the tree further – might as well assume that further symbols have been selected randomly.

- further j^{th} transmitted symbol is equally likely to be a 0 or 1.

For an $[n_0, k_0=1]$ CC a path of depth d $\Rightarrow n_i = n_0 d$

Pr[of this path] = Pr[information sequence generating it]
= (1/2) .....(1/2) $= 2^{-d} = p_i$
d times

$$\therefore \quad \frac{1}{n_i} log_2\left(\frac{1}{p_i}\right) = \frac{1}{n_0 d} log_2 2^d = \frac{1}{n_0} = \text{R code rate.}$$

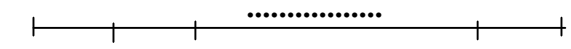
$$\therefore \quad \mu(\hat{\mathbf{x}}_i) = \sum_{j=1}^{n_i} \underbrace{\left\{ log_2 \frac{\Pr[y_i \mid x_{ij}]}{\Pr[y_i]} - R \right\}}$$

Fano metric $\Rightarrow$ assumed channel is DMC.
if different metric could change.

Problems with Stack Algorithm Implementation

1) Erasure (Loss of data)
   - need input buffer to more incoming data
   - long searches can cause input buffer to over flow; i.e., data erasure.
   ~ Probability of erasure 10^{-3} common.
   - essentially independent of encoder memory.
   $\Rightarrow$ choose codes with large memory.
   $\Rightarrow$ (*) multiple stack algorithm
   (Chevillat and Costello) – eliminates erasures.
2) Finite Stack Size – when fills up
   - delete path at the bottom.
3) Stack Reordering
   - Stack bucket algorithm – Jelinek.

Quantize metric range
(+21 to -110 for example)



Each new path is entered as the top entry in the bucket containing its metric – no reordering.

$\Rightarrow$ Not always is the "best" path extended just a very good path.

FANO algorithm – another approach

Distance Measures for CC

Definition – l^{th} minimum distance d_l^* of a CC is the smallest Hamming distance between any 2 initial codeword segments l frames long that disagree in the initial frame.

d_{M+1}^* called the minimum distance $\equiv d^*$

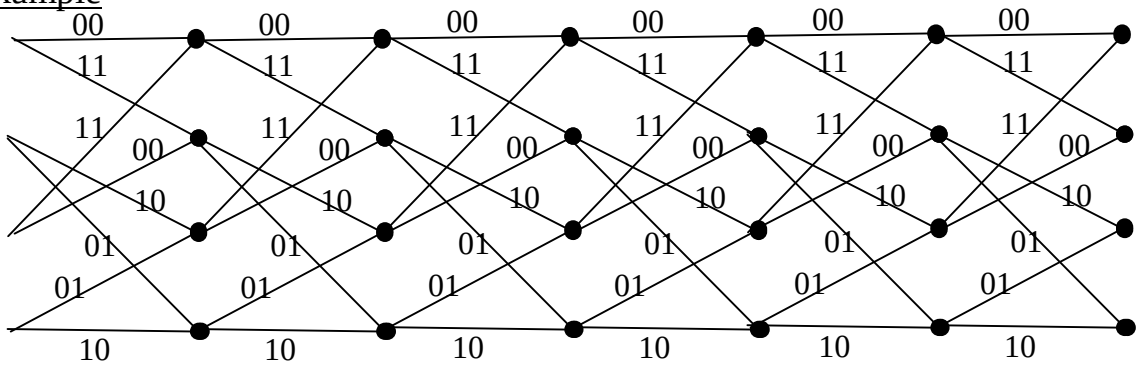
Sequence $d_1^* \leq d_2^* \leq d_3^* \leq d_4^* \leq$ ..... distance profile

Free distance $d_f = \max_l d_l^* = \lim_{l\to\infty} d_l^* = d_\infty$

Definition – Free length, n_∞ , of a CC is the length of the nonzero segment of a smallest – weight CC of nonzero weight.

n_∞ = length at which $d_l^* = d_\infty$

Example



distance profile – CC linear $\Rightarrow$ 1 of the codewords can be taken as the all zero paths.

$d_1^* = 2$, $d_5^* = 4$
$d_2^* = 3$, $d_6^* = 5$
$d_3^* = 3$, :
$d_4^* = 4$, $\therefore d_f = d_6^* = 5$
$n_\infty = 6$

For this CC if 2 or fewer error in the 1st 6 frames then the errors in the 1st frame can be corrected.

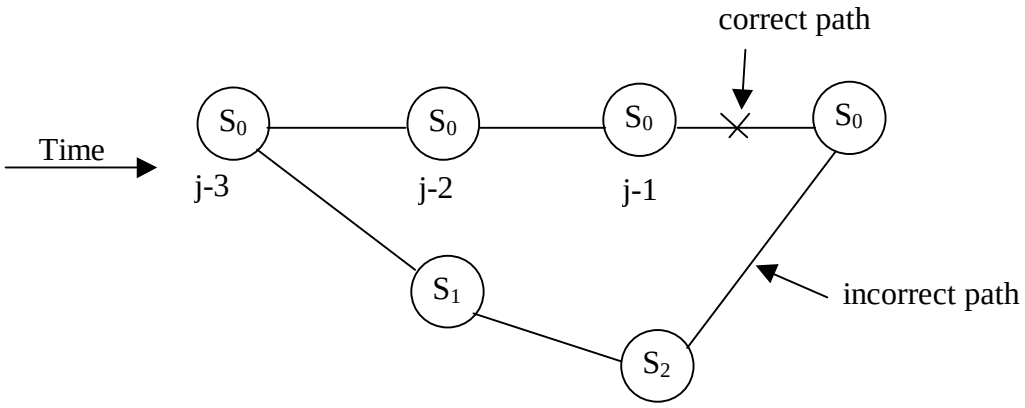
Error Performance

Performance Bounds for Convolutional Codes
Performance of VA on a BSC

Example
$G(x) = [1+x, 1+x^2, 1+x+x^2]$
Assume without loss of generality, all-zero codeword is transmitted.

$$T(D, L, I) = \frac{D^7L^3I}{1 - DLI(1 + D^2L)}$$
$$= D^7L^3I + D^8L^4I^2+D^9L^5I^3+D^{10}(L^5I^2+L^6I^4)+.....$$

i.e. one codeword (a path in trellis) of weight 7 length 3 generated by an info. sequence of weight 1, one path of weight 8 and length 4, generated by weight 2 info sequence, and so on.



First-event error at time unit j

- The path must be one of the paths enumerated by T(D, L, I). If it is the weight 7 path, a first event error will be made, the binary sequence (recovered) **r** agree with the incorrect path in four or more positions.

If BSC transition probability is p

$$P_7 = P[\text{4 or more 1's in seven positions}]$$
$$= \sum_{l=4}^{7} \binom{7}{l} p^l (1-p)^{7-l}$$

If the weight 8 path is the incorrect path, a first event error is made with probability.

$$P_8 = \frac{1}{2}\binom{8}{4}p^4(1-p)^4 + \sum_{l=5}^{8}\binom{8}{l}p^l(1-p)^{8-l}$$

Since if the metrics of the correct path and incorrect paths are tied, an error is made with probability ½.
In general, if the incorrect path has weight d, a first event error is made with probability

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

Since all incorrect paths of length j branches or less can cause a first event error at time unit j, the <u>first event error probability</u> at time unit j, $P_f(E, j)$ can be overbounded, using a union bound. If all incorrect paths of length greater than j branches are also included there.

$$P_f(E, j) < \sum_{d=d_{free}}^{\infty} A_d P_d \qquad \leftarrow \text{independent of j}$$

where A_d is the number of paths with weight d.

$$\Rightarrow \quad P_f(E) < \sum_{d=d_{free}}^{\infty} A_d P_d \qquad \leftarrow \text{first event error probability at any time unit}$$

For d odd

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

It can also be shown that (*) is an upper bound on P_d for d even.

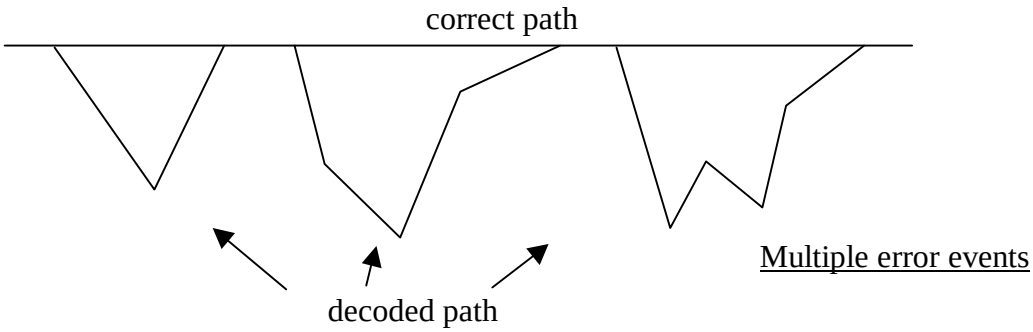
$$\Rightarrow \quad P_f(E) < \sum_{d=d_{free}}^{\infty} A_d \left[2\sqrt{p(1-p)}\right]^d$$

For an arbitrary convolutional code with transfer function

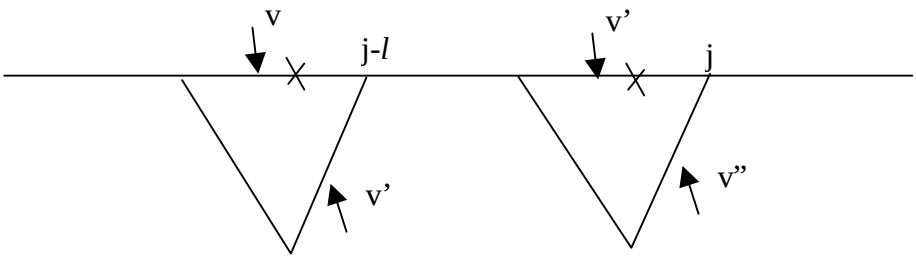
$$T(D) = \sum_{d=d_{free}}^{\infty} A_d D^d$$

$$P_f(E) < T(D)\Big|_{D=2\sqrt{p(1-p)}}$$

- The final decoded path can diverge from and emerge from the correct path any number of times.



After one or more errors have occurred, the two paths compared at all zero state will both be incorrect paths.



We say that an event error occurs at time j, if v" survives over v'.
The bound derived earlier can be applied to event error probability (independent of j).

$$P(E) < \sum_{d=d_{free}}^{\infty} A_d P_d < T(D)\Big|_{D=2\sqrt{p(1-p)}}$$

For small p, the bound is dominated by the 1st term (i.e., the free distance term)

$$\Rightarrow \quad P(E) \approx A_{d_{free}} \left[2\sqrt{p(1-p)}\right]^{d_{free}}$$

$$\approx A_{d_{free}} 2^{d_{free}} p^{d_{free}/2}$$

Example

For the earlier code $d_{free} = 7$, $A_{d_{free}} = 1$ for $p = 10^{-2}$

$$P(E) \approx 2^7 p^{7/2} = 1.28 \cdot 10^{-5}$$

- The event error probability bound can be modified to provide a bound on the bit error probability, $P_b(E)$, i.e., the expected number of information bit decoding errors per decoded information bit.
- Each error event causes a number of information bit errors equal to the number of nonzero information bits on the incorrect path.
- Hence if each event error probability term P_d is weighted by the number of nonzero bits on the weight d path, or if there are more than one weight d path, by the total number nonzero information bit decoding errors made at any time results.

It can then be derived by k, the number of information bits per unit time to bound $P_b(E)$

$$\Rightarrow \quad P_b(E) < \frac{1}{k} \sum_{d=d_{free}}^{\infty} B_d P_d$$

where B_d is the total number of nonzero information bits on all weight d paths. Since

$$T(d, L, I) = \sum_{d=d_{free}}^{\infty} \sum_{b=1}^{\infty} A_{d,b} D^d I^b$$

It follows that

$$\frac{\partial T(D,I)}{\partial I}\Big|_{I=1} = \sum_{d=d_{free}}^{\infty} \sum_{b=1}^{\infty} b A_{d,b} D^d = \sum_{d=d_{free}}^{\infty} B_d D^d$$

where $B_d = \sum_{b=1}^{\infty} b A_{d,b}$

$$\Rightarrow \quad P_b(E) < \frac{1}{k} \sum_{d=d_{free}}^{\infty} B_d \left[2\sqrt{p(1-p)}\right]^d = \frac{1}{k} \frac{\partial T(D,I)}{\partial I}\Big|_{D=2\sqrt{p(1-p)},\, I=1}$$

For an arbitrary CC with T(D, I)
For small p the bound is dominated by the 1st term,

$$P_b(E) \approx \frac{1}{k} B_{d\,free} \left[2\sqrt{p(1-p)}\right]^{d_{free}}$$
$$\approx \frac{1}{k} B_{d\,free} 2^{d_{free}} p^{d_{free}/2}$$

Example
Earlier code ; $B_{d\,free}$ =1, d_{free} = 7, p = 10^{-2}

$$P_b(E) = 2^7 p^{7/2} = 1.28 \,.\, 10^{-5}$$

Same as P(E), i.e., when p small, the most likely error event is that the weight 7 path is decoded instead of all zero thereby causing one information bit error. Typically then, each error even causes are bit error.

If the BSC is derived from an AWGN channel with BPSK modulation, optimum coherent detection, and binary output quantization, then

$$p = Q\left(\sqrt{\frac{2E}{N_0}}\right)$$

where E is the energy per transmitted symbol and N_0 is the (one-sided) noise PSD.

$$p \approx \frac{1}{2} e^{-E/N_0}$$

and for a CC with d_{free}

$$P_b(E) \approx \frac{1}{k} B_{d\,free} 2^{d_{free}/2} e^{-(d_{free}/2)(E/N_0)}$$

when p is small (i.e. when E/N_0 is large). Defining energy per info. bit E_b as

$$E_b \equiv \frac{E}{R}$$

Since 1/R is the number of transmitted symbols per info. bit we can write,

$$P_b(E) \approx \frac{1}{k} B_{d_{free}} 2^{d_{free}/2} e^{-(Rd_{free}/2)(E_b/N_0)} \text{ (with coding)} \qquad \text{(A)}$$

for large E_b/N_0.

On the other hand if no coding is used (R=1), the BSC transition probability is the bit error probability $P_b(E)$ and

$$P_b(E) \approx \frac{1}{2} e^{-E_b/N_0} \text{ (without coding)} \qquad \text{(B)}$$

Comparing (A), (B), the (negative) exponent with coding is larger by a factor of $Rd_{free}/2$ then the exponent without coding. Since the exponential term dominates the error probability expressions for large E_b/N_0,

$$\gamma \equiv 10 \log_{10} \frac{Rd_{free}}{2} \text{ dB}$$

is called <u>asymptotic coding gain</u> in the hard decision case.

- Coding gains become smaller as E_b/N_0 becomes smaller. If E_b/N_0 is reduced to the point where code rate R is greater than the channel capacity C, reliable communication with coding is no longer possible, and an uncoded system will outperform a coded system.

- <u>For a binary input AWGN with no output quantization</u>

$$P(E) < T(D)\Big|_{D=e^{-RE_b/N_0}}$$

$$P_b(E) < \frac{1}{k} \frac{\partial T(D,I)}{\partial I}\Big|_{D=e^{-RE_b/N_0},\, I=1}$$

$$\approx \frac{1}{k} B_{d_{free}} \left(e^{-RE_b/N_0}\right)^{d_{free}}$$

$$= \frac{1}{k} B_{d_{free}} e^{-Rd_{free}E_b/N_0} \qquad \text{(**)}$$

Compare this with BSC case, the exponent is larger by a factor of 2.

$\Rightarrow$ <u>This is equivalent to a 3-dB energy (or power) advantage for the AWGN channel over the BSC</u>

⇒ Illustrate the benefits of allowing an unquantized demodulatior output instead of making hard decisions.

- The decoder complexity increases due to the need to accept soft i/ps.
- Over the entire range of E_b/N_0 ratios, the decibel loss associated with hard decisions runs between 2-3 dB. Hence the use of soft decisions (Q>2 but still finite) has become increasingly popular recently to regain the most of 2→3 dB loss due to hard quantization while keeping the advantages of digital decoding.

Soft Decision Decoding of Convolutional Codes

(k_0 , n_0) Convolutional code ,

BPSK Signaling, Output symbol Energy = E_s

AWGN Channel with noise variance $\sigma^2 = N_0 / 2$

The symbols transmitted can be considered independent of each other. Therefore,

$$p(\mathbf{R}|\mathbf{Y}) = \prod_{k=1}^{N}\prod_{j=1}^{n_0} p(r_j^k \mid y_j^k)$$

R is the received sequence corrupted with AWGN and **Y** is the transmitted sequence.

$$p(r_j^k \mid y_j^k) = \frac{1}{\sqrt{pN_0}} \exp\left[-\frac{\left(r_j^k - y_j^k\sqrt{E_s}\right)^2}{N_0}\right] \text{ where } y_j^k \text{ take the values +1 or –1.}$$

Substituting in the above equation and taking logarithm we have,

$$\text{Log } p(\mathbf{R}|\mathbf{Y}) = \sum_{k=1}^{N}\left\{\sum_{j=1}^{n_0}\left[-\frac{\left(r_j^k - y_j^k\sqrt{E_s}\right)^2}{N_0} - \log\sqrt{pN_0}\right]\right\}$$

$$= \frac{-1}{N_0} \sum_{k=1}^{N} \left\{ \sum_{j=1}^{n_0} \left(r_j^k - y_j^k \sqrt{E_s} \right)^2 \right\} - \frac{N n_0}{2} \log p N_0$$

This is equivalent to finding the minimum Euclidean distance path through the trellis.

Neglecting constants and since $\left(y_j^k\right)^2 = 1$, the branch metric is given by

$\sum_{j=1}^{n_0} r_j^k y_j^k$ and the individual bit metrics are

$M(r_j^k \mid y_j^k) = r_j^k y_j^k.$

In practice, usually some quantized version of the received signal is processed.