Lecture Notes 17

Goals

- Be able to encode using a convolutional code
- Be able to decode a convolutional code received over a binary symmetric channel or an additive white Gaussian channel

Convolutional Codes

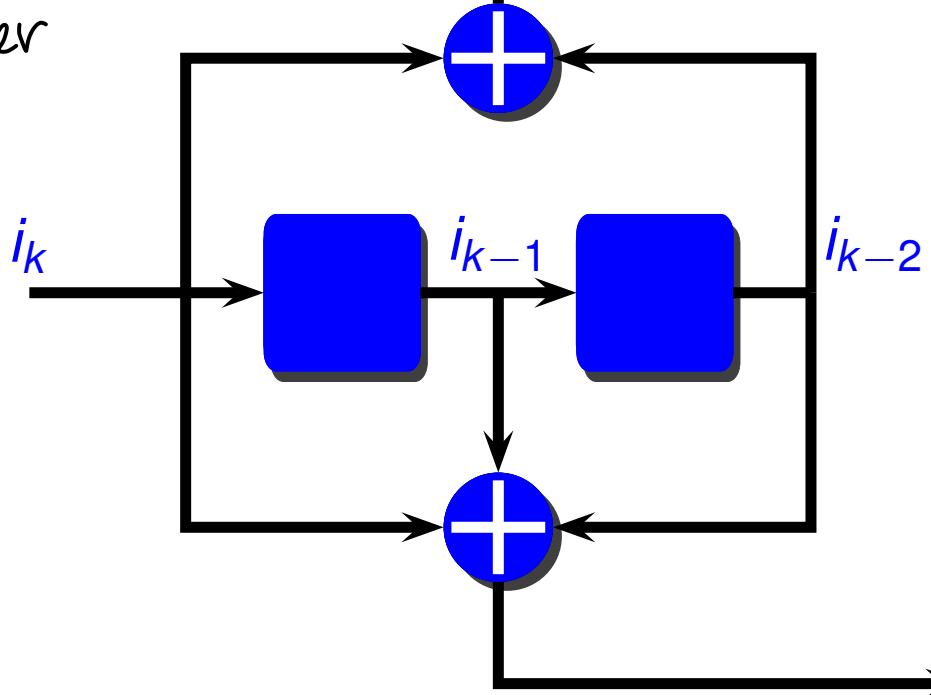
- Convolutional codes are an alternative way of introducing redundancy into the data stream.
- They are preferred to block codes in many situations because of the ease with which soft decision decoding can be performed.
- They are called convolutional code because the encoder output can be written as the convolutional of the encoder input with a generator sequence.
- A convolutional code consists of k shift registers of length M or less each.
- The n outputs are linear combinations of the contents of the shift registers and the input.
- A convolutional code is describe by the memory length M (or constraint length $K = M + 1$), the number of input bits k the number of output bits n and the connections between the shift registers and the outputs.

Convolutional Codes: Example ($K = 3, M = 2$, rate 1/2 code)

mod 2

$$c_k^{(0)} = i_k + i_{k-2}$$

shift register



$$c_k^{(1)} = i_k + i_{k-1} + i_{k-2}$$

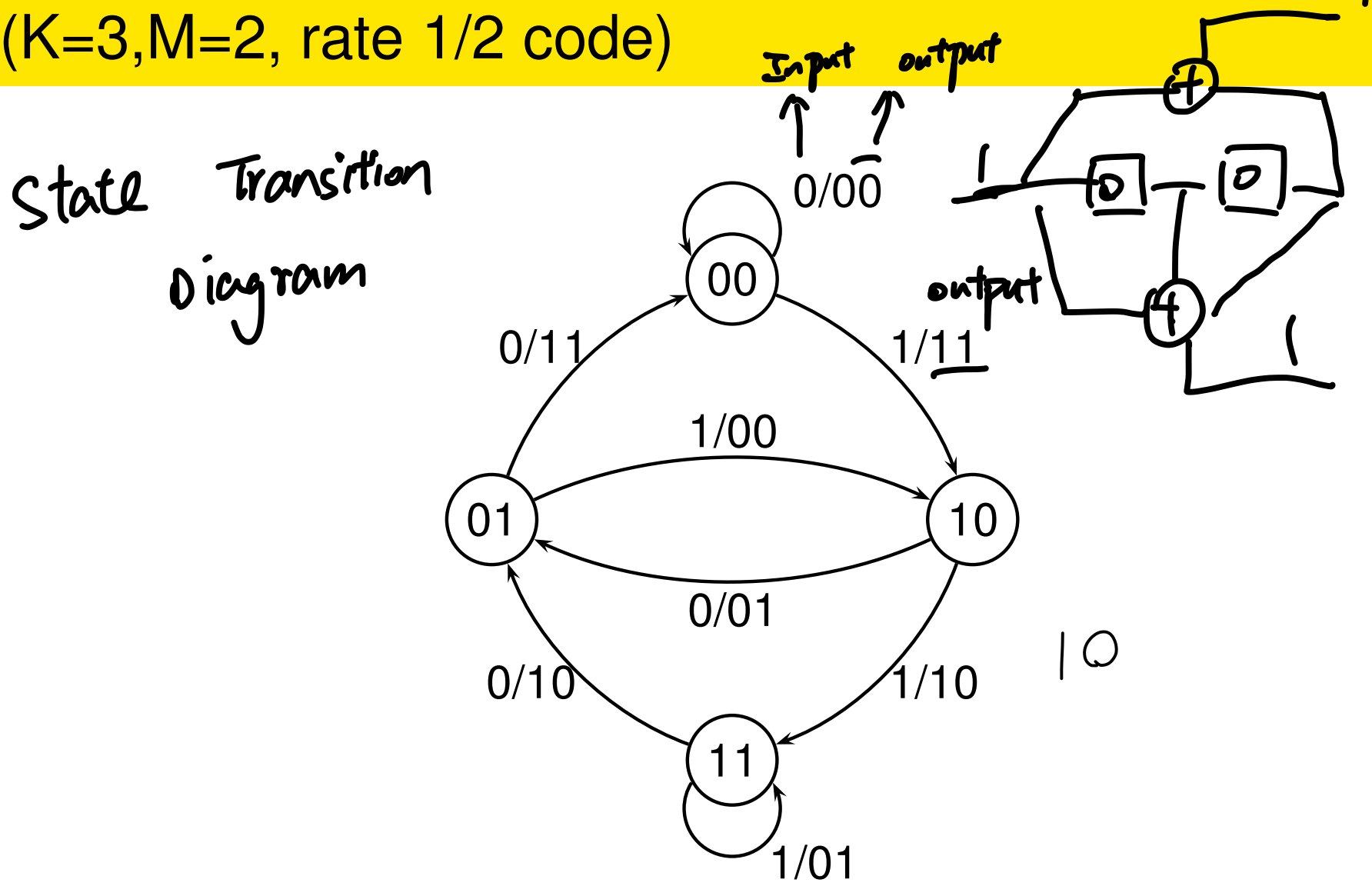
Convolutional Codes: Example ($K = 3, M = 2$, rate 1/2 code)

$$
\begin{aligned}
c_k^{(0)} &= i_k + i_{k-2} \\
&= g_0^{(0)} i_{k-0} + g_1^{(0)} i_{k-1} + g_2^{(0)} i_{k-2} \\
&= \sum_{l=0}^{2} g_l^{(0)} i_{k-l} \qquad \text{convolution} \\
c_k^{(1)} &= i_k + i_{k-1} + i_{k-2} \\
&= g_0^{(1)} i_{k-0} + g_1^{(1)} i_{k-1} + g_2^{(1)} i_{k-2} \\
&= \sum_{l=0}^{2} g_l^{(1)} i_{k-l}
\end{aligned}
$$

$$
g_0^{(0)} = 1,\ g_1^{(0)} = 0,\ g_2^{(0)} = 1
$$

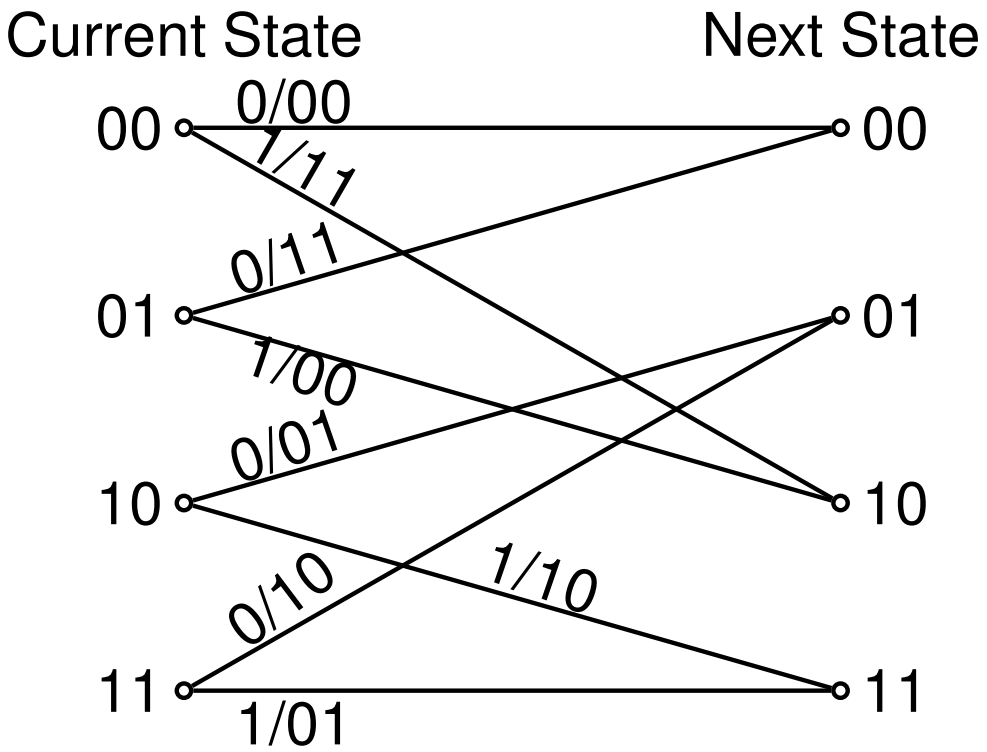
$$
g_0^{(1)} = 1,\ g_1^{(1)} = 1,\ g_2^{(1)} = 1
$$

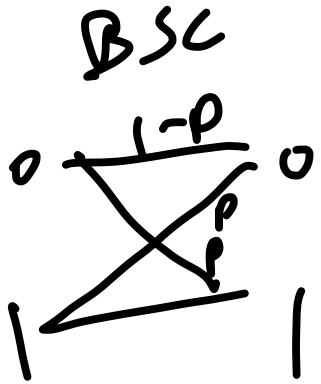
State Diagram for Convolutional Codes: Example (K=3,M=2, rate 1/2 code)



Trellis Diagram

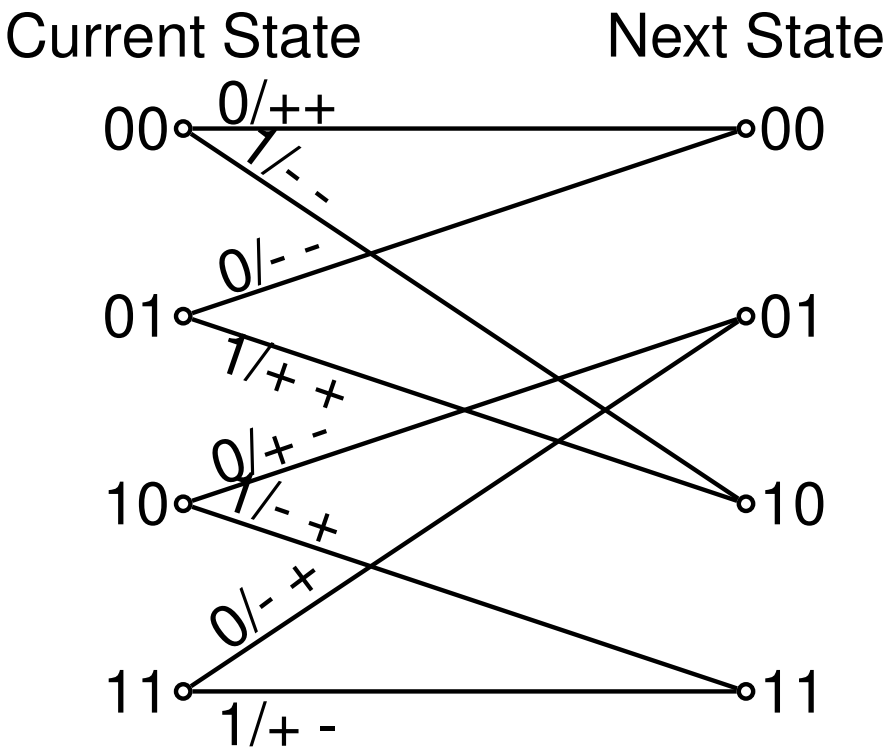
We can describe the sequence of states the encoder passes through in time via a trellis diagram. This will be useful for decoding purposes.

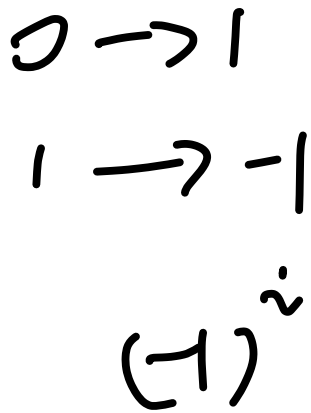




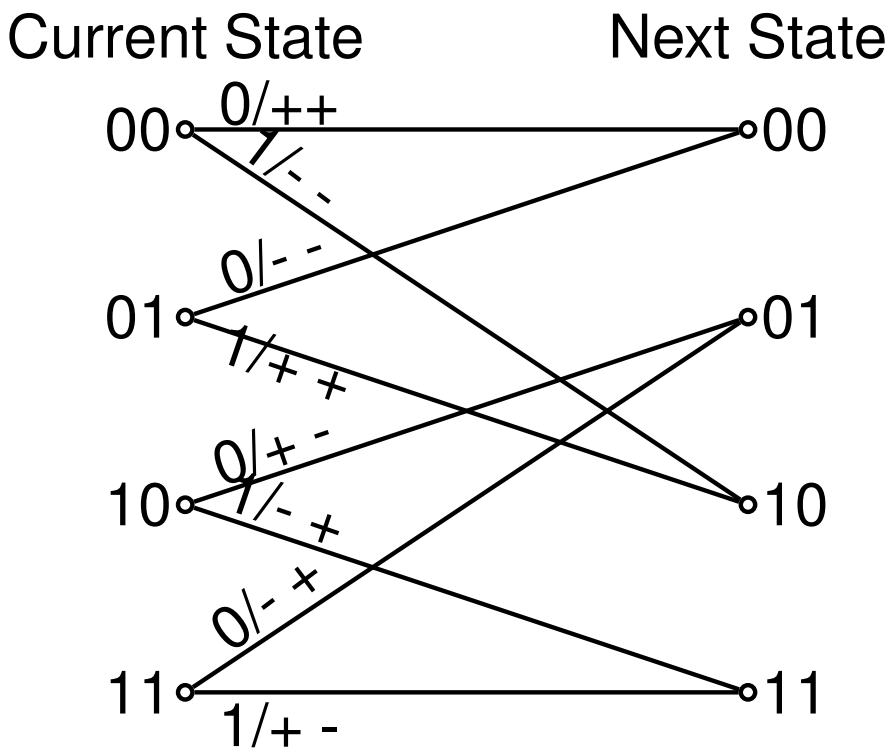
Trellis Diagram

For decoding when the modulation is BPSK and the channel is an additive white Gaussian noise channel it is better to use the ± 1 representation of the coded symbols. In this case the trellis diagram can be relabeled.

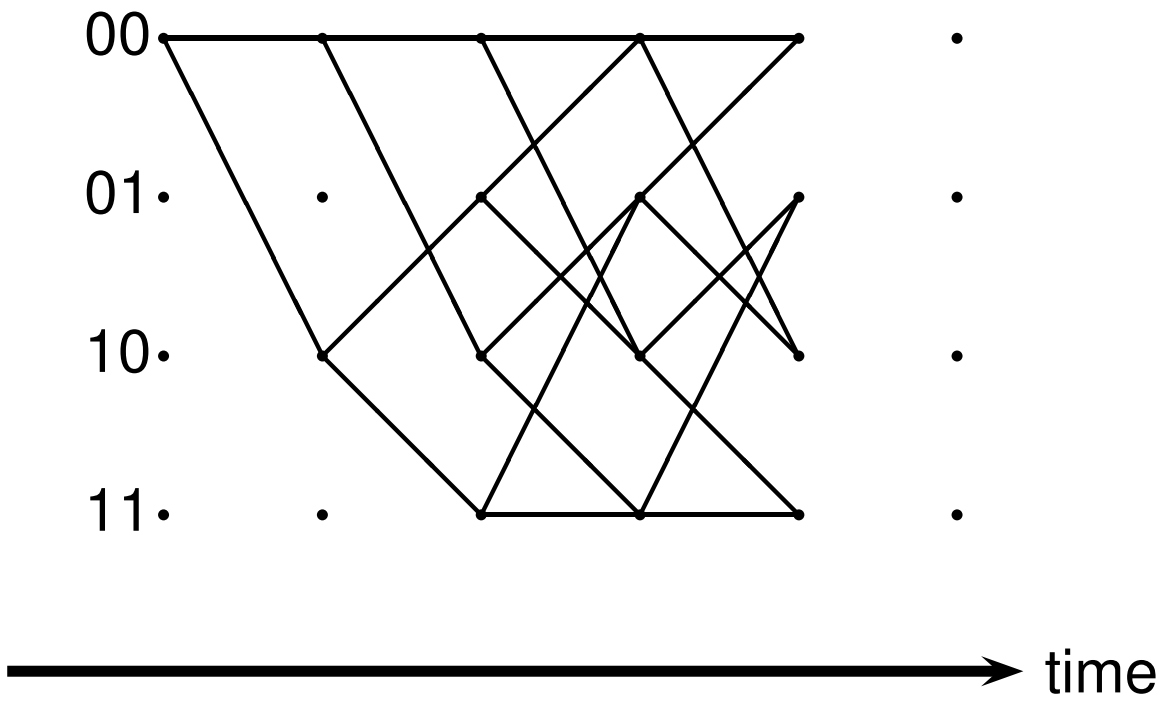




Trellis



Trellis Diagram



Decoding Convolutional Codes

Consider a state diagram for a particular convolutional code. Let x_m be the sequence representing the state at time m. Let x_0 be the initial state of the process $(p(x_0) = 1)$. Later on we will denote the states by the integers 1,2,...,N. Since this is a Markov process we have that

$$p(x_{m+1}|x_m, x_{m-1}, ..., x_1, x_0) = p(x_{m+1}|x_m)$$

That is, the state at time $m+1$ depends only on the state at time m and not any previous state.
Let $w_m = (x_{m+1}, x_m)$ be the state transition at time m.

Decoding Convolutional Codes

There is a one-to-one correspondence between state sequences and transition sequences. By some mechanism (e.g. a noisy channel) a noisy version $\{z_m\}$ of the state transition sequence is observed. Based on this noisy version of $\{w_m\}$ we wish to estimate the state sequence $\{x_m\}$ or the transition sequence w_m. Since $\{w_m\}$ and $\{x_m\}$ contain the same information we have that

$$p(\mathbf{z}|\mathbf{x}) = \mathbf{p}(\mathbf{z}|\mathbf{w})$$

where $\mathbf{z} = z_0, z_1, ..., z_{M-1}$, $\mathbf{x} = x_0, x_1, ..., x_M$, and $\mathbf{w} = w_0, ..., w_{M-1}$. If the channel is memoryless then we have that

$$p(\mathbf{z}|\mathbf{w}) = \prod_{\mathbf{m=0}}^{\mathbf{M-1}} \mathbf{p}(\mathbf{z_m}|\mathbf{w_m})$$

Decoding Convolutional Codes

So given an observation $\mathbf{z}$ find the state sequence $\mathbf{x}$ for which the a posteriori probability $p(\mathbf{x}|\mathbf{z})$ is largest. This minimizes the probability that we chose the wrong sequence.
Thus the optimum (minimum sequence error probability) decoder chooses $\mathbf{x}$ which maximizes $p(\mathbf{x}|\mathbf{z})$: i.e

$$\begin{aligned}\hat{\mathbf{x}} &= \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x}|\mathbf{z}) \\ &= \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x},\mathbf{z}) \\ &= \operatorname{argmin}_{\mathbf{x}} \{-\log p(\mathbf{x},\mathbf{z})\} \\ &= \operatorname{argmin}_{\mathbf{x}} \{-\log p(\mathbf{z}|\mathbf{x})\mathbf{p}(\mathbf{x})\}\end{aligned}$$

Decoding Convolutional Codes

Using the memoryless property of the channel we obtain

$$p(\mathbf{z}|\mathbf{x}) = \prod_{\mathbf{m=0}}^{\mathbf{M-1}} \mathbf{p}(\mathbf{z_k}|\mathbf{w_m})$$

and using the Markov property of the state sequence

$$p(\mathbf{x}) = \left[\prod_{m=0}^{M-1} p(x_{m+1}|x_m)\right] p(x_0)$$

Define $\lambda(w_m)$ as follows:

$$\lambda(w_m) = -\ln p(x_{m+1}|x_m) - \ln p(z_m|w_m)$$

Then

$$\hat{\mathbf{x}} = \text{argmin}_{\mathbf{x}}\{\sum_{m=0}^{M-1} \lambda(w_m)\}$$

Decoding Convolutional Codes

This problem formulation leads to a recursive solution. The recursive solution is called the Viterbi Algorithm by communication engineers and is a form of Dynamic Programming as studied by control engineers. They are really the same though.

Decoding Convolutional Codes

VITERBI ALGORITHM

Let $\Gamma(x_m)$ be the length (optimization criteria) of the shortest (optimum) path to state x_m at time m. Let $\hat{\mathbf{x}}(x_m)$ be the shortest path to state x_m at time m. Let $\hat{\Gamma}(x_{m+1}, x_m)$ be the length of the path to state x_{m+1} at time m that goes through state x_m at time m.

Then the algorithm works as follows.

Storage

m, time index,

$\hat{\mathbf{x}}(x_m)$, $x_m \in \{1, 2, ..., M\}$

$\Gamma(x_m)$, $x_m \in \{1, 2, ..., M\}$

Initialization

$m = 0$,

o $\hat{\mathbf{x}}(x_0) = x_0$

$\hat{\mathbf{x}}(x_m)$ arbitrary, $m \neq x_0$

$\Gamma(x_0) = 0$

$\Gamma(x_m) = \infty$, $m \neq 0$

Decoding Convolutional Codes

Recursion

$\hat{\Gamma}(x_{m+1}, x_m) = \Gamma(x_m) + \lambda(w_m)$

$\Gamma(x_{m+1}) = \min_{x_m} \hat{\Gamma}(x_{m+1}, x_m)$ for each x_{m+1}

Let $\hat{\mathbf{x}}_m(x_{m+1}) = \operatorname{argmin}_{x_m} \hat{\Gamma}(x_{m+1}, x_m)$.

$\hat{\mathbf{x}}(x_{m+1}) = [\hat{\mathbf{x}}(x_m), x_{m+1}]$

Justification of Decoding Convolutional Codes

- Basically we are interested in finding the shortest length path through the trellis.
- At time m we find the shortest length paths to each of the possible states at time m by computing all possible ways of getting to state $x_m = u$ from a state at time $m-1$.
- If the shortest path (denoted by $\hat{x}(u)$) to get to $x_m = u$ at time m goes through state $x_{m-1} = v$ at time $m-1$ (i.e. $\hat{\mathbf{x}}(u) = \hat{\mathbf{x}}(v), u$) then the corresponding path $\hat{\mathbf{x}}(v)$ to state $x_{m-1} = v$ must be the shortest path to state v at time $m-1$ since if there was a shorter path, say $\tilde{x}(v)$, to state v at time $m-1$ then the path $\tilde{\mathbf{x}}(v), u$ to state u at time m that used this shorter path to state v at time $m-1$ would be shorter then what we assumed was the shortest path).

Justification of Decoding Convolutional Codes

- Stated another way if the shortest way of getting to state u at time m is by going through state v at time $m-1$ then the path used to get to state v at time $m-1$ must be the shortest of all paths to state v at time $m-1$.

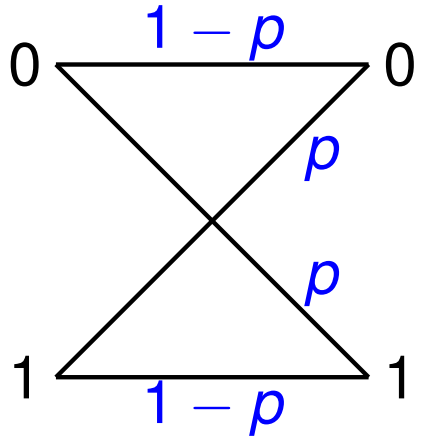
Decoding a Convolutional Code over a BSC

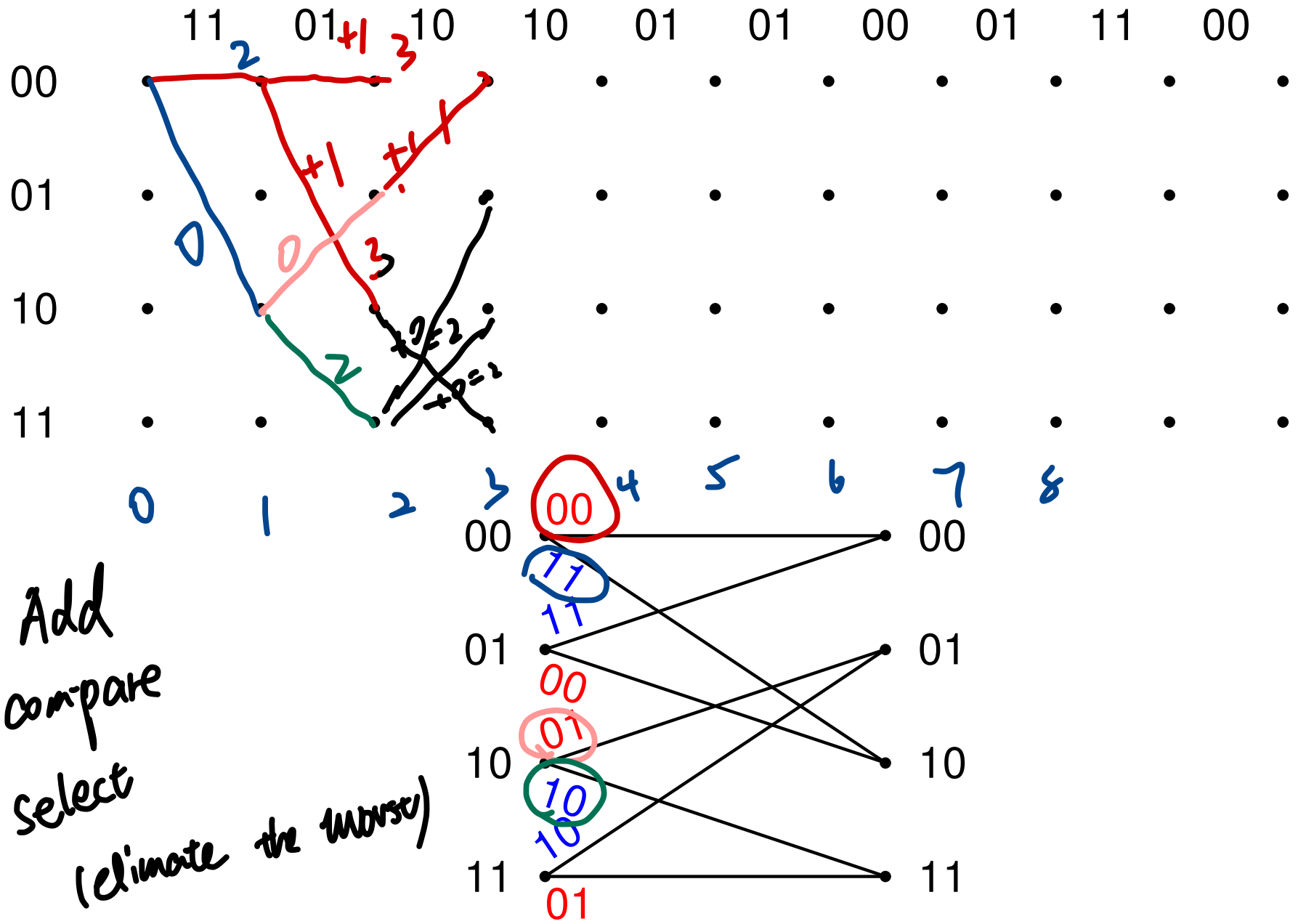
- Suppose an information sequence of length k bits is the input to a convolutional encoder.
- The encoded sequence is transmitted over a binary symmetric channel.
- The received sequence contains errors.
- The decoder wants to find the information sequence that corresponds to a coded sequence closest in Hamming distance to the received sequence.

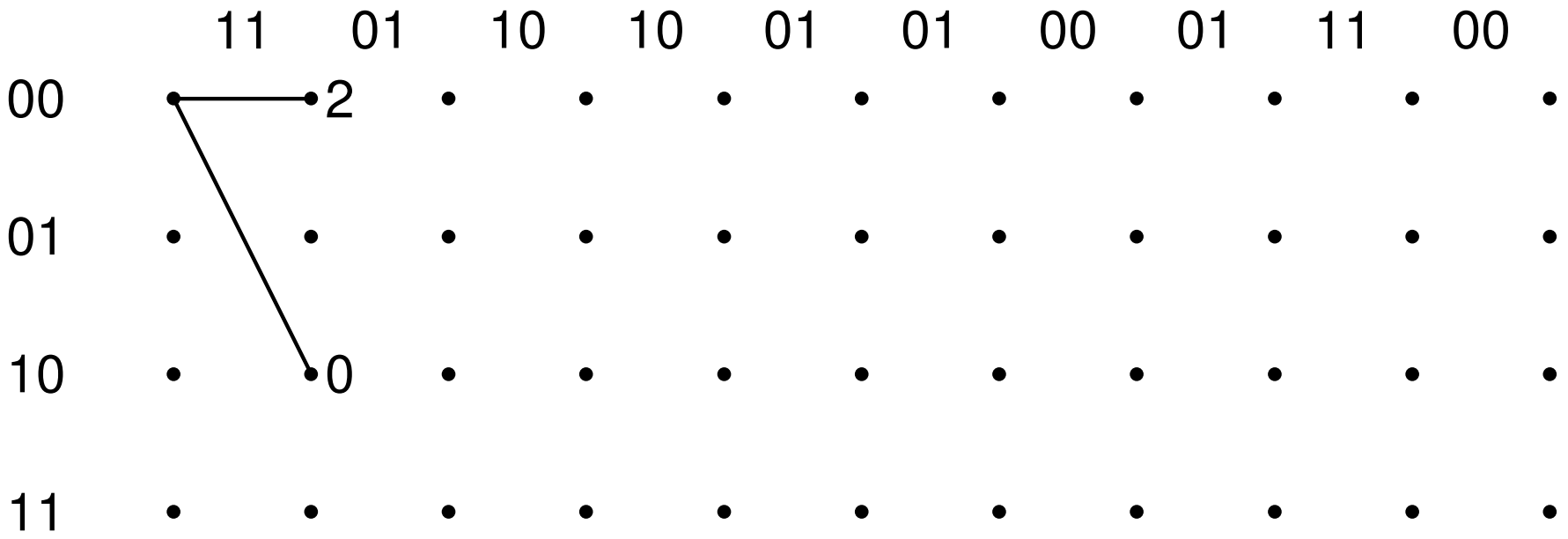
BIBO – AWGN

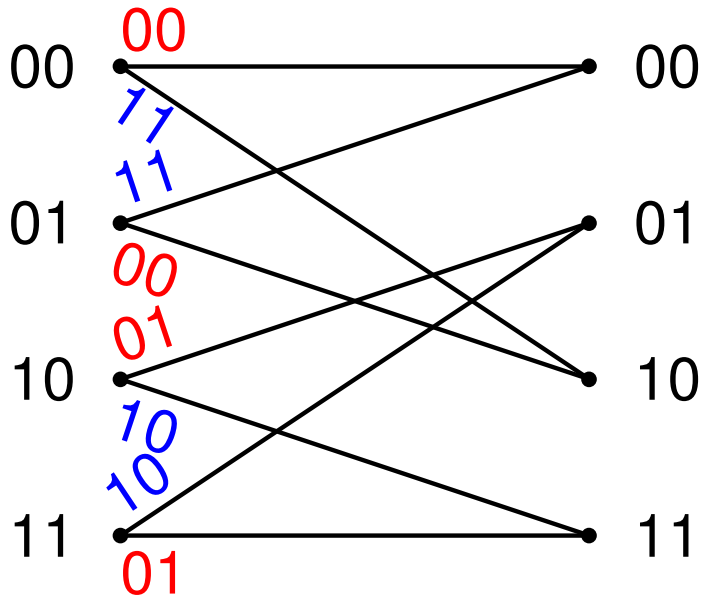
$p = Q\left(\sqrt{\frac{2E}{N_0}}\right)$

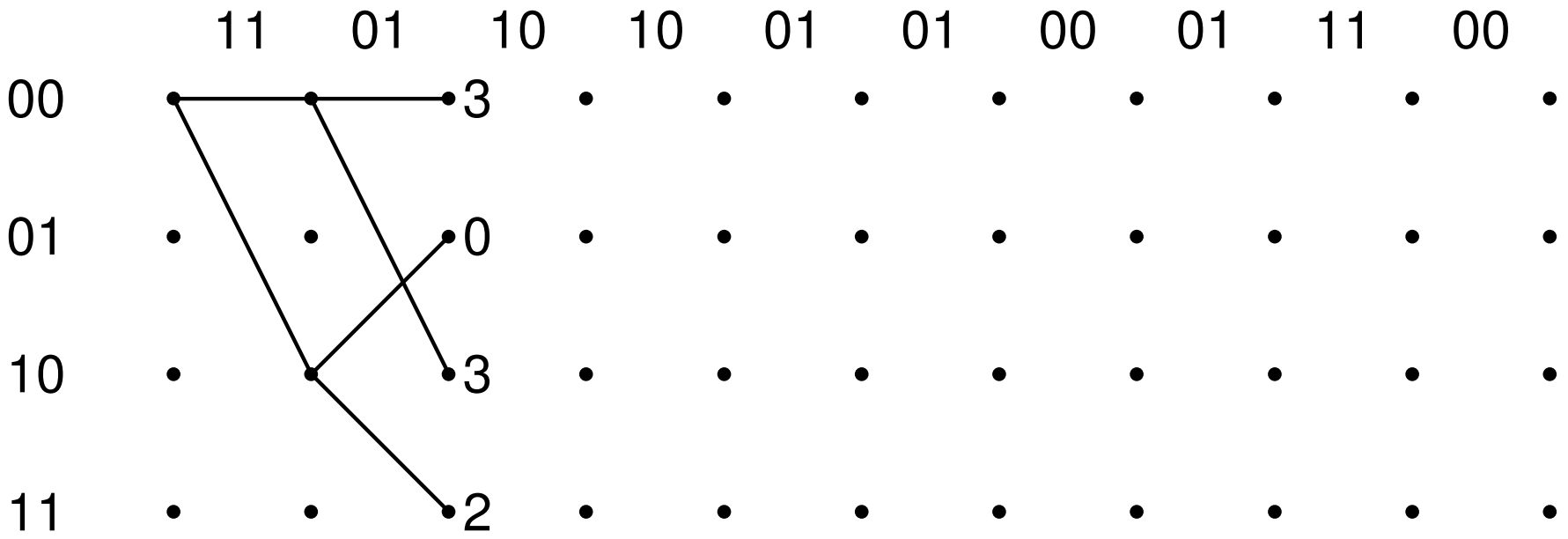
$p < \frac{1}{2}$

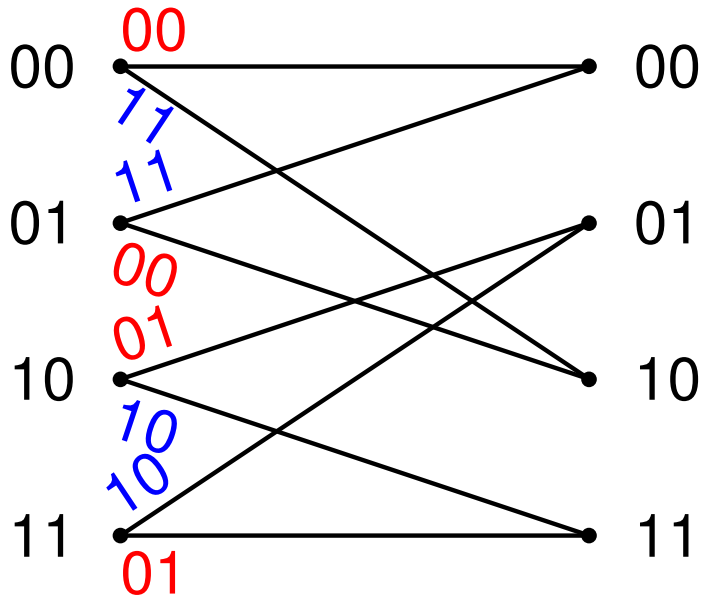


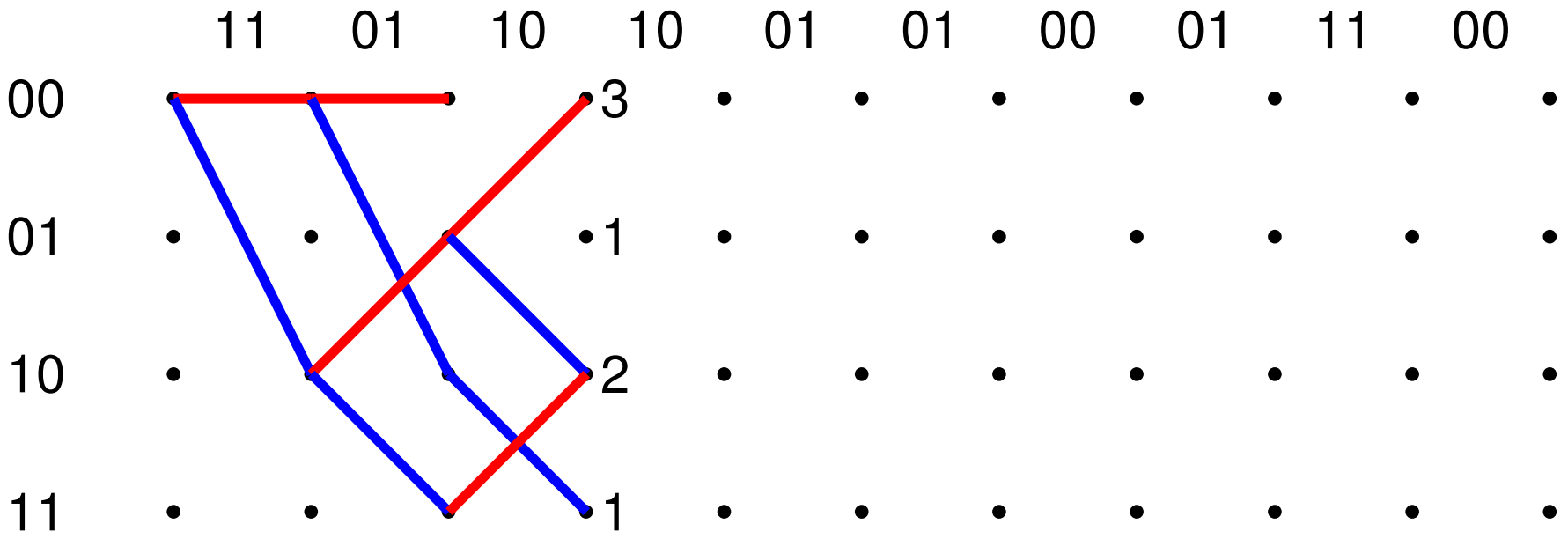

11
01
10
10
01
01
00
01
11
00
00
01
10
11
2
+1
3
+1
+1
0
0
3
2
2
2
0
1
2
3
4
5
6
7
8
00
11
11
00
01
10
10
00
01
10
11
01
00
01
10
11
Add
compare
select
(eliminate the worse)

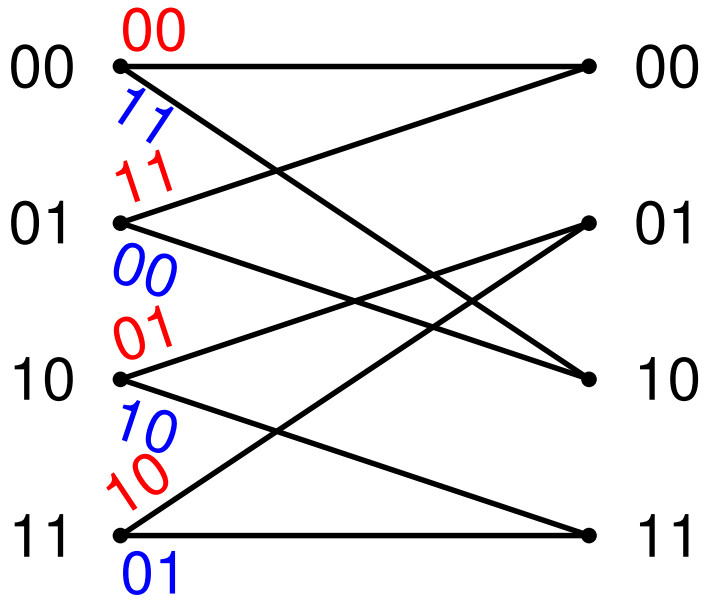

11 01 10 10 01 01 00 01 11 00
00
01
10
11
2
0

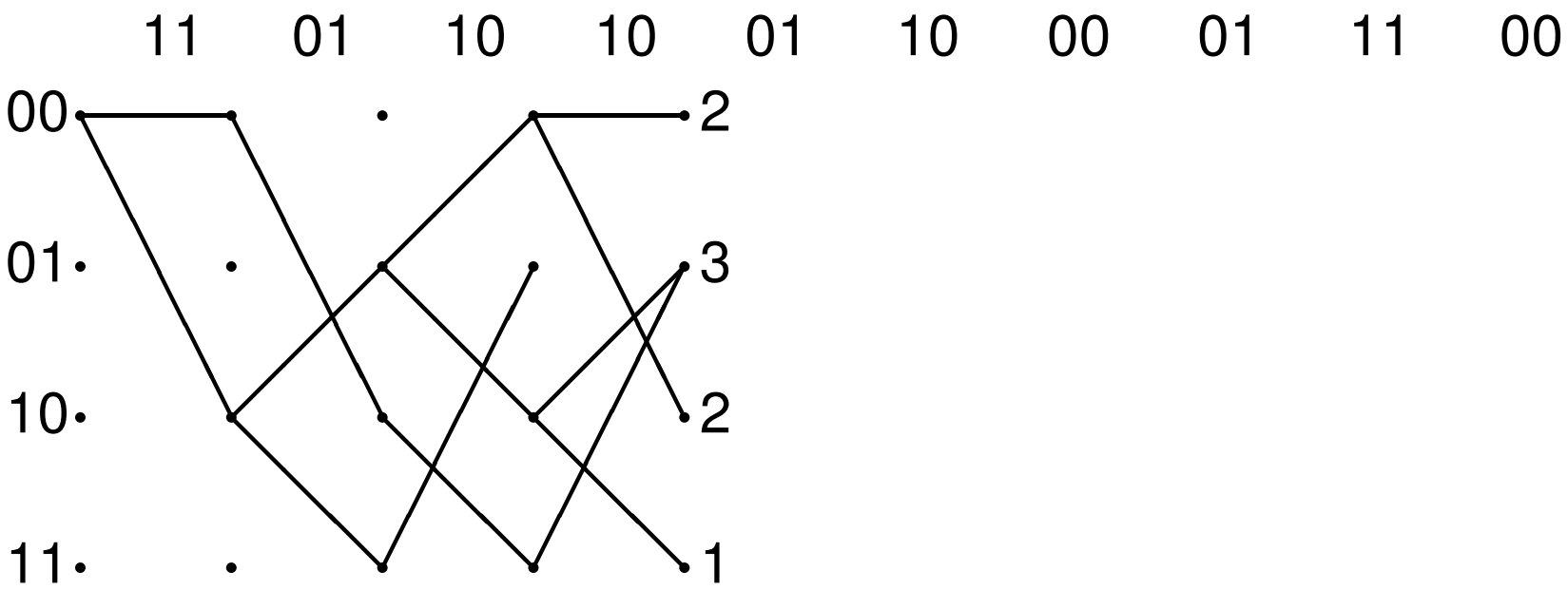

00
01
10
11
00
01
10
11
00
11
11
00
01
10
10
01

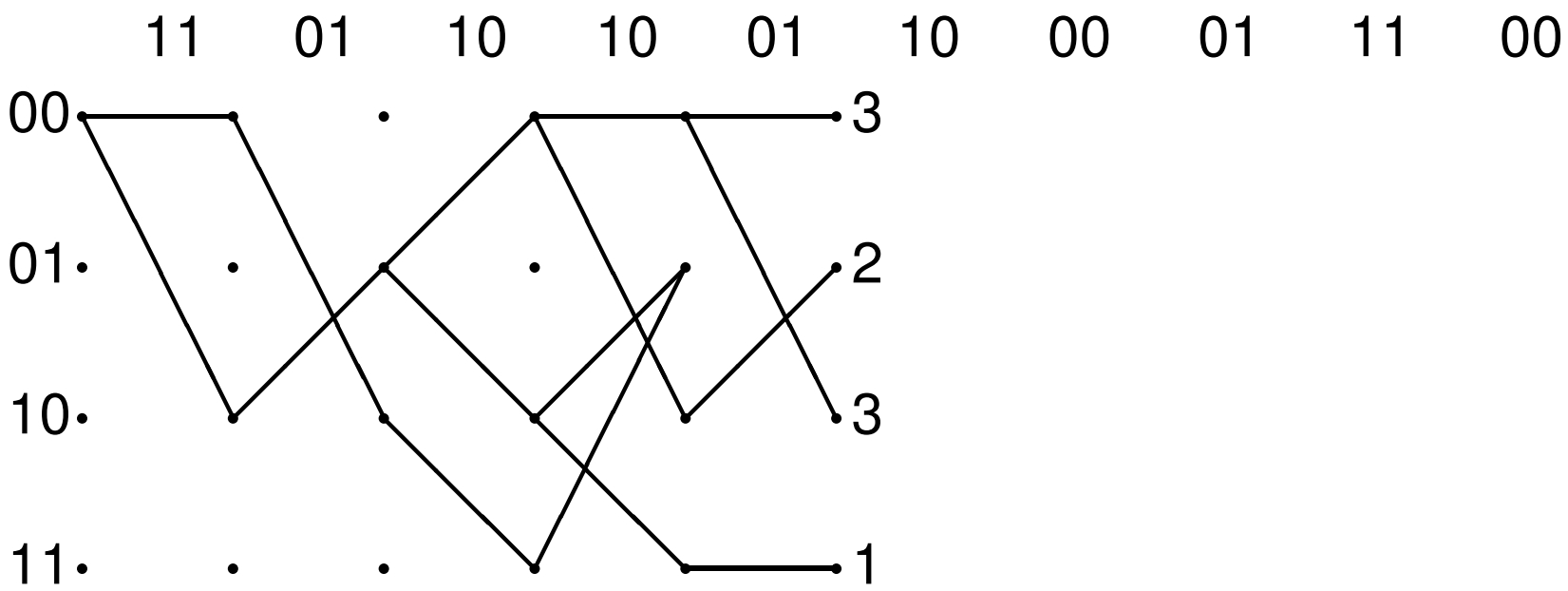

11 01 10 10 01 01 00 01 11 00
00
01
10
11
3
0
3
2

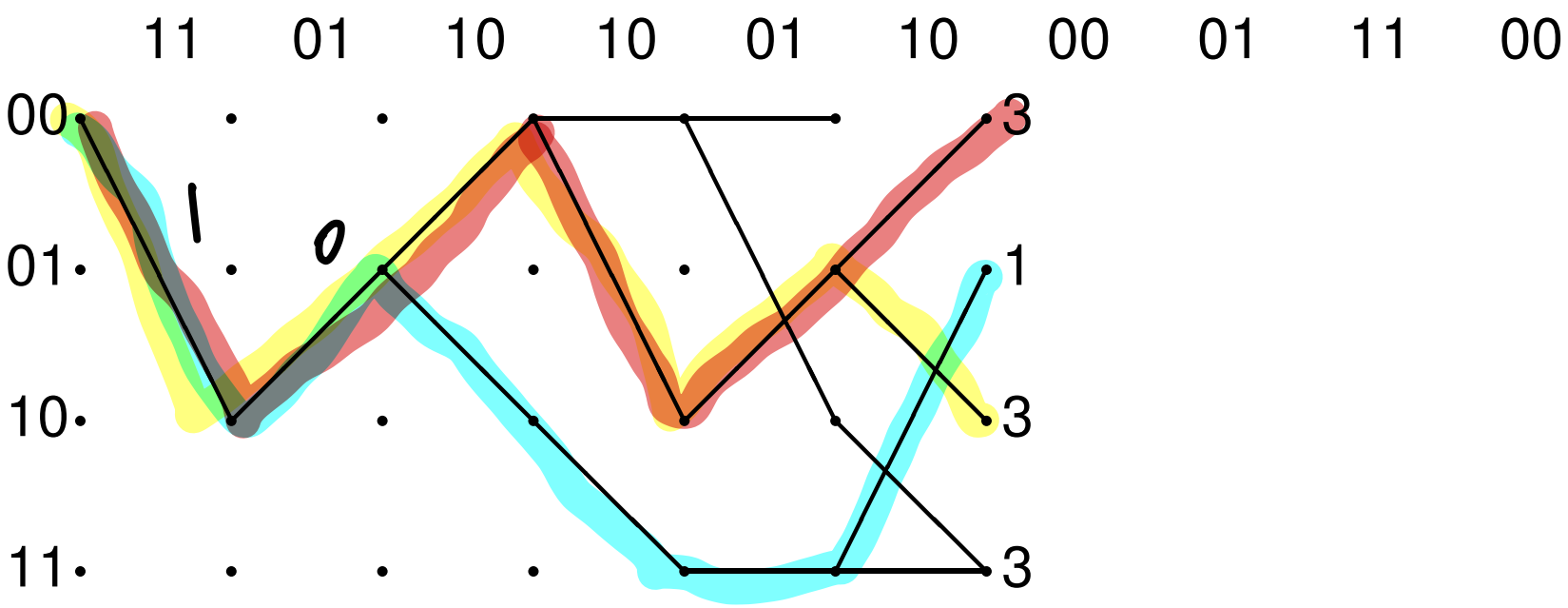

00
01
10
11
00
11
11
00
01
10
10
01
00
01
10
11

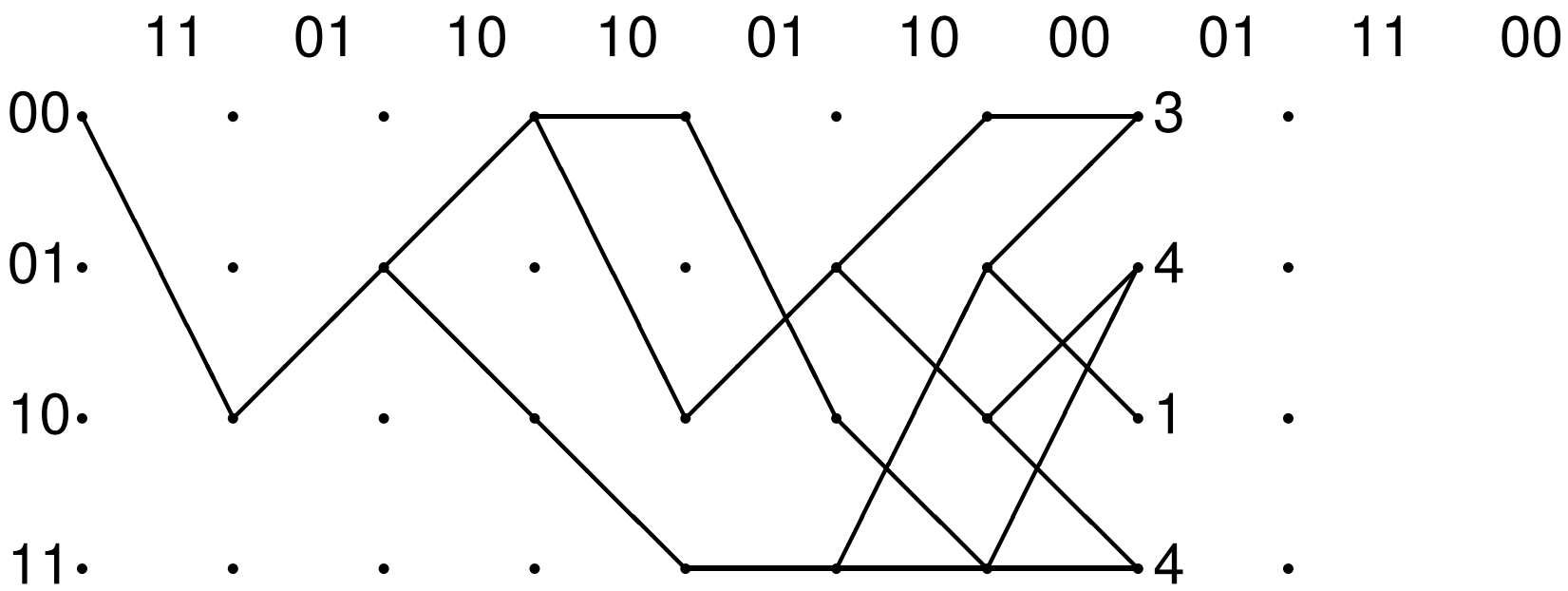

11 01 10 10 01 01 00 01 11 00
00
01
10
11
3
1
2
1

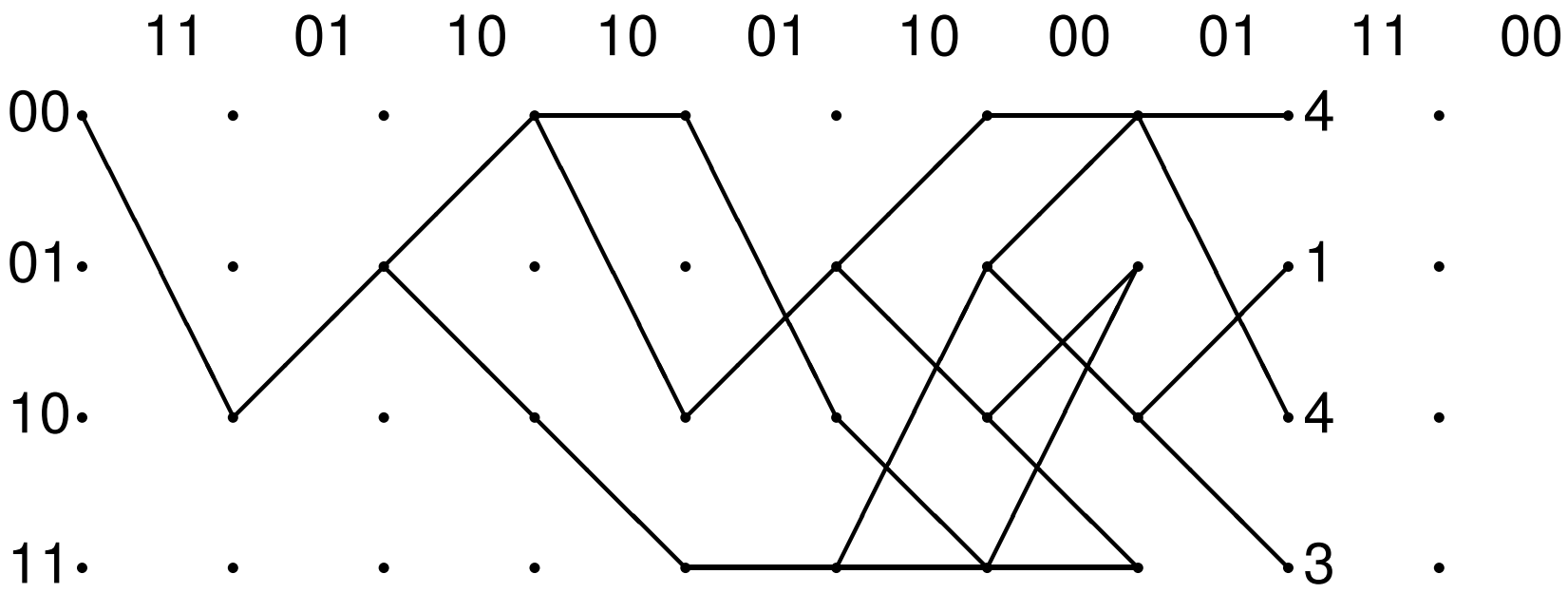

00
01
10
11
00
11
11
00
01
10
10
01
00
01
10
11

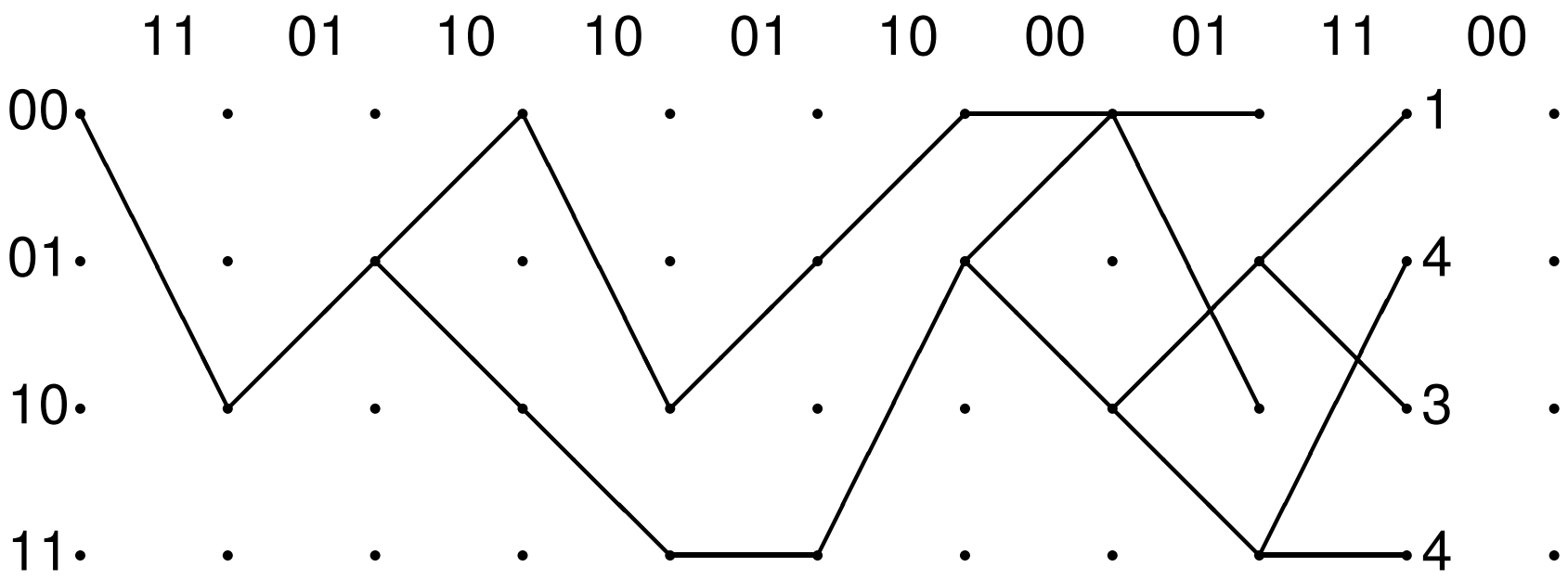

11 01 10 10 01 10 00 01 11 00
00
01
10
11
2
3
2
1

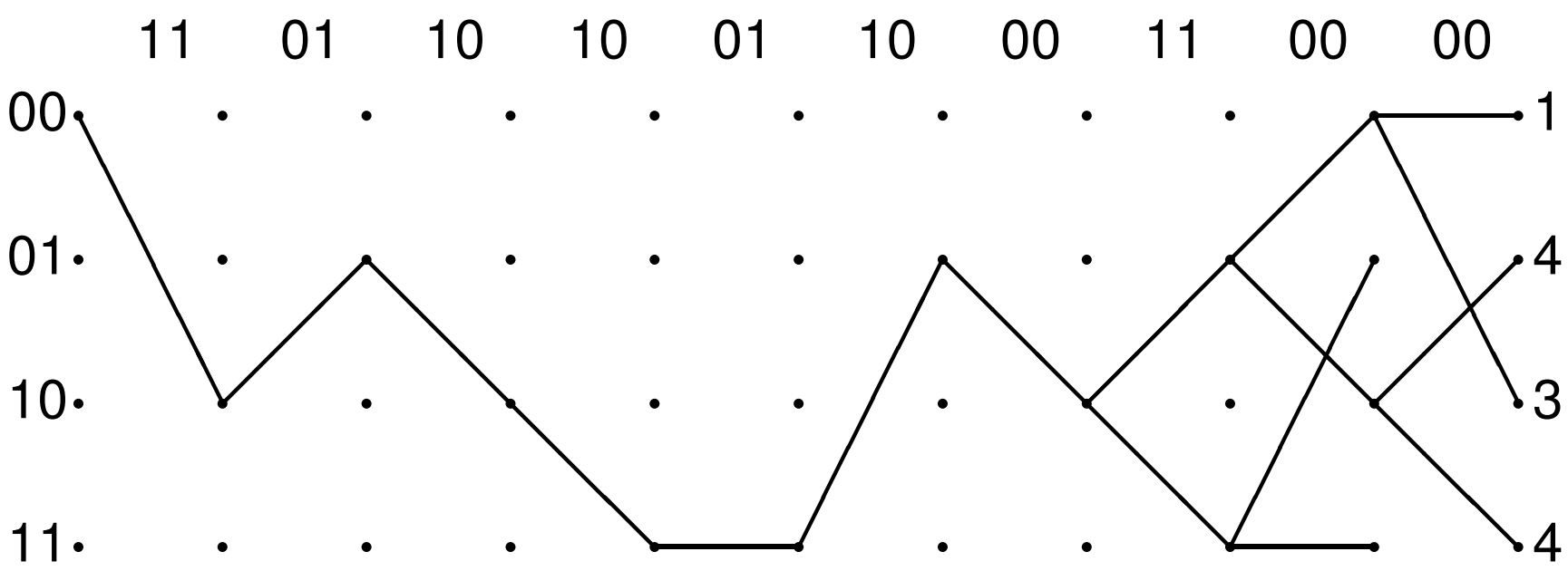

11 01 10 10 01 10 00 01 11 00
00
01
10
11
3
2
3
1


11 01 10 10 01 10 00 01 11 00
00
01
10
11
3
1
3
3
1
0


11 01 10 10 01 10 00 01 11 00
00
01
10
11
3
4
1
4


11 01 10 10 01 10 00 01 11 00
00
01
10
11
4
1
4
3


11 01 10 10 01 10 00 01 11 00
00
01
10
11
1
4
3
4


11 01 10 10 01 10 00 11 00 00
00
01
10
11
1
4
3
4

Truncation of Convolutional Codes

- Usually a finite number of bits are transmitted at a time.
- It is generally a good idea to make sure the encoder of the convolutional code starts and ends in the all zero state.
- To do this we append a few ’0’ bits at the end to force the encoder back into the all zero state.
- In decoding then the decoder knows after a certain number of received symbols that the encoder input was ’0’ and eliminates paths that do not correspond to a ’0’ input.

Example of Truncation of Convolutional Codes

$r = \frac{1}{2}$

- Consider the same example as before.
- Now suppose that the encoder consisted of 6 information bits followed by 2 tail bits.
- The encoder would then produce 16 coded bits.
- The effective rate of communication then would be 6 information bits/ 16 coded bits.

100 into bits + 2 tail bits = 102

⇓

204 output bits

$\text{rate} = \frac{100}{204} \approx \frac{1}{2}$

Decoding a Convolutional Code over an AWGN

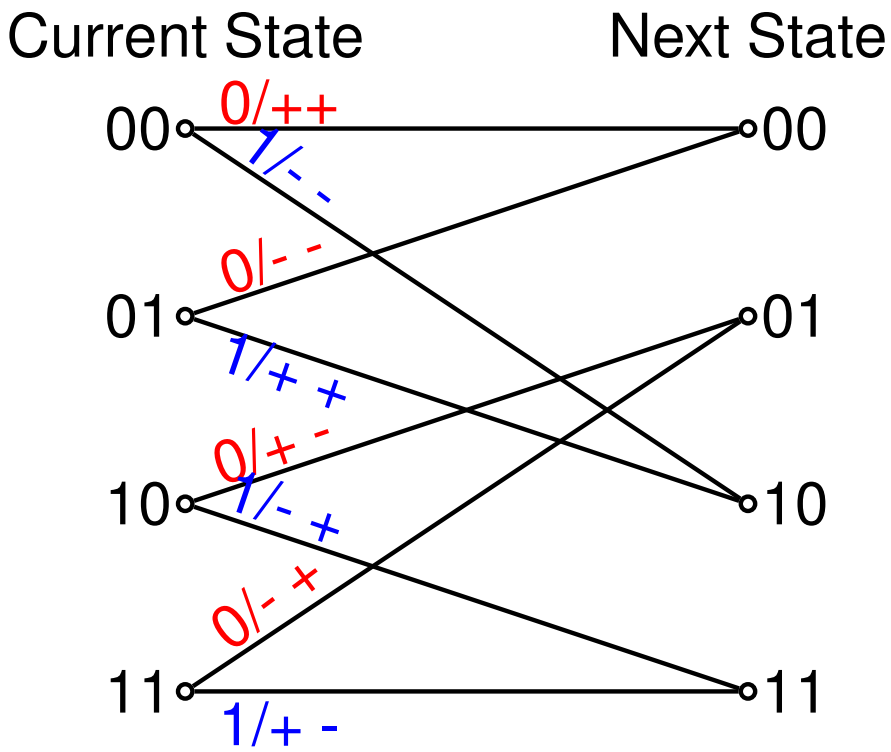
BI - AWGN

- Suppose an information sequence of bits is the input to a convolutional encoder. The coded bits are mapped to ± 1 before transmission.
- The encoded sequence is transmitted over an additive white Gaussian noise channel.
- The received sequence contains noise.
- The decoder wants to find the information sequence that corresponds to a coded sequence closest in (squared) Euclidean distance to the received sequence.

If have all codeword the same energy, find largest correlation.

Trellis Diagram

For decoding when the modulation is BPSK and the channel is an additive white Gaussian noise channel it is better to use the ± 1 representation of the coded symbols. In this case the trellis diagram can be relabeled.

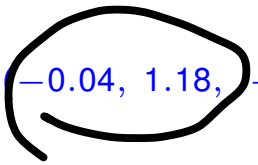


Decoding a Convolutional Code over an AWGN

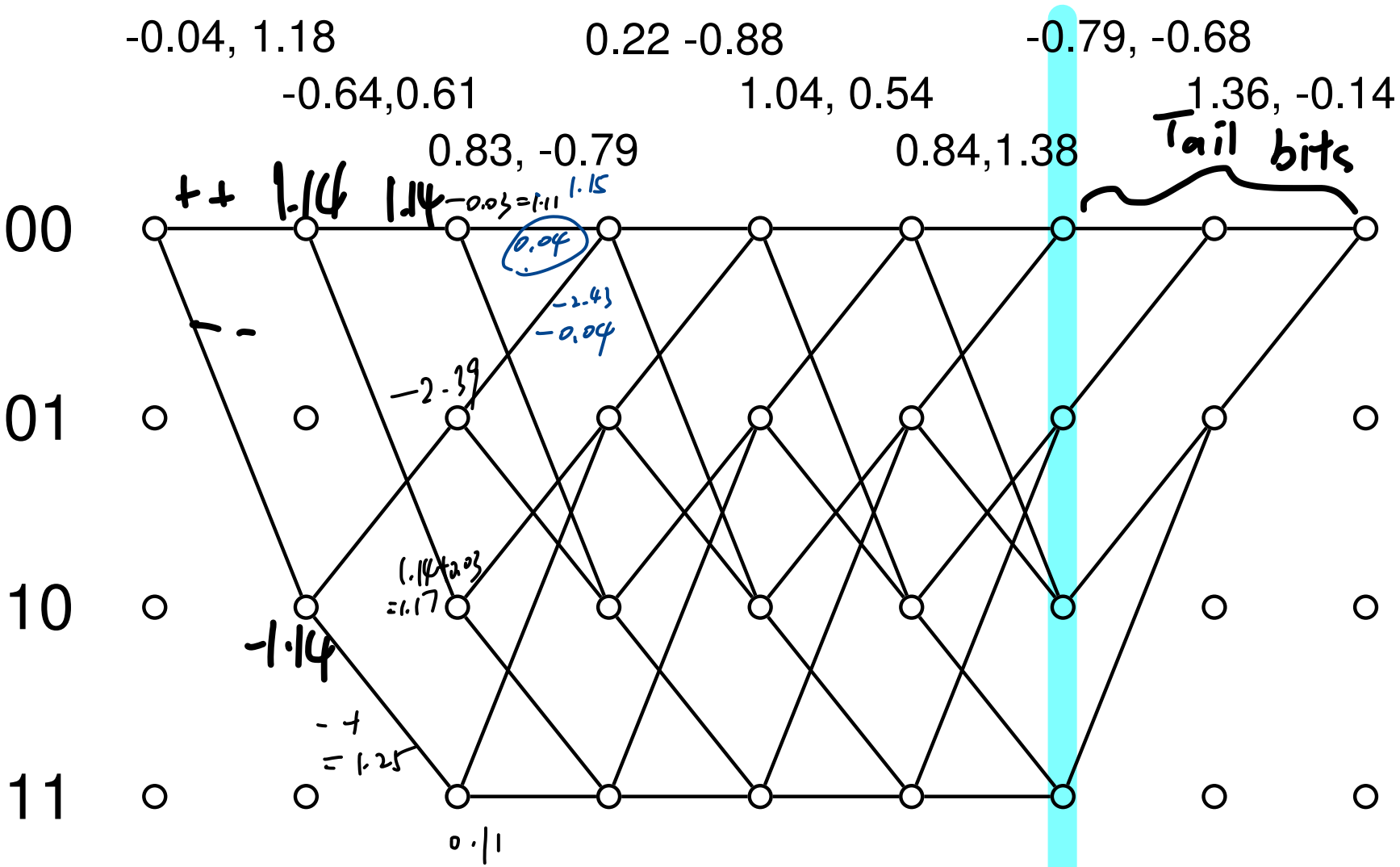
Transmit = $(-1, 1, \cdots 1)$

Suppose the received vector was

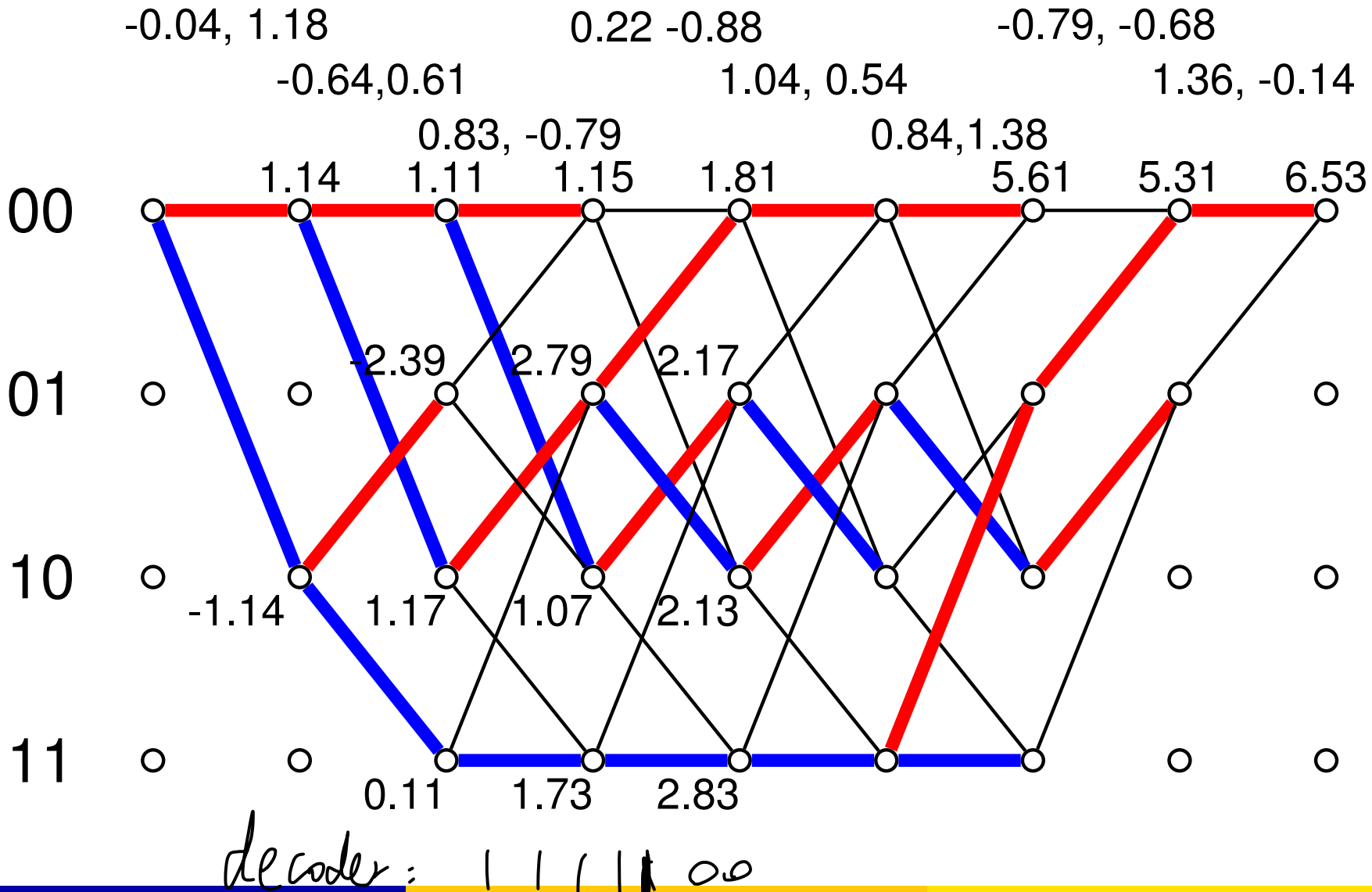
$$r = (-0.04,\ 1.18,\ -0.64,\ 0.61,\ 0.83,\ -0.79,\ 0.22\ -0.88,\ 1.04,\ 0.54,\ 0.84,\ 1.38,\ -0.79,\ -0.68\ 1.36,\ -0.14).$$



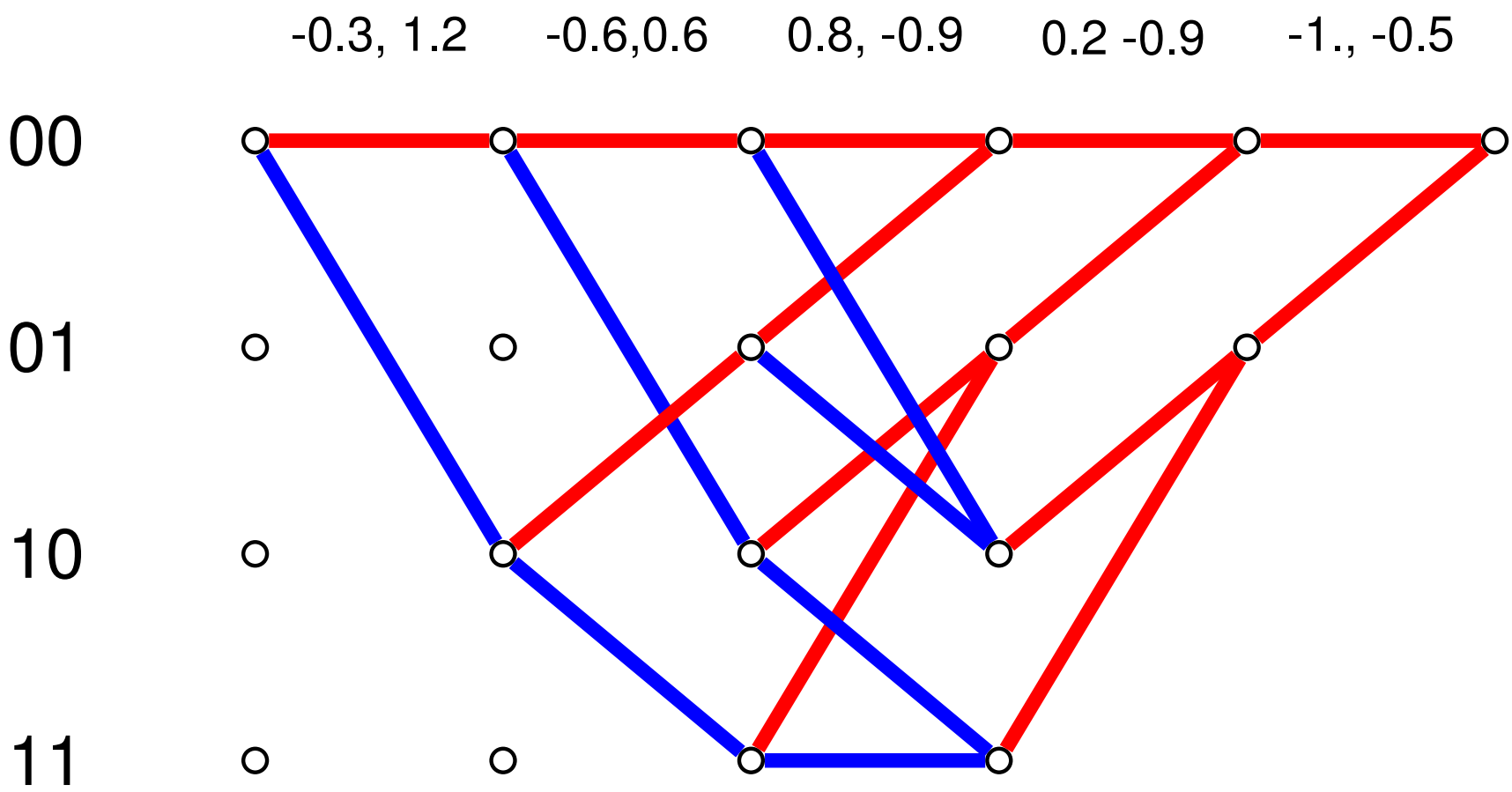
Truncated Trellis



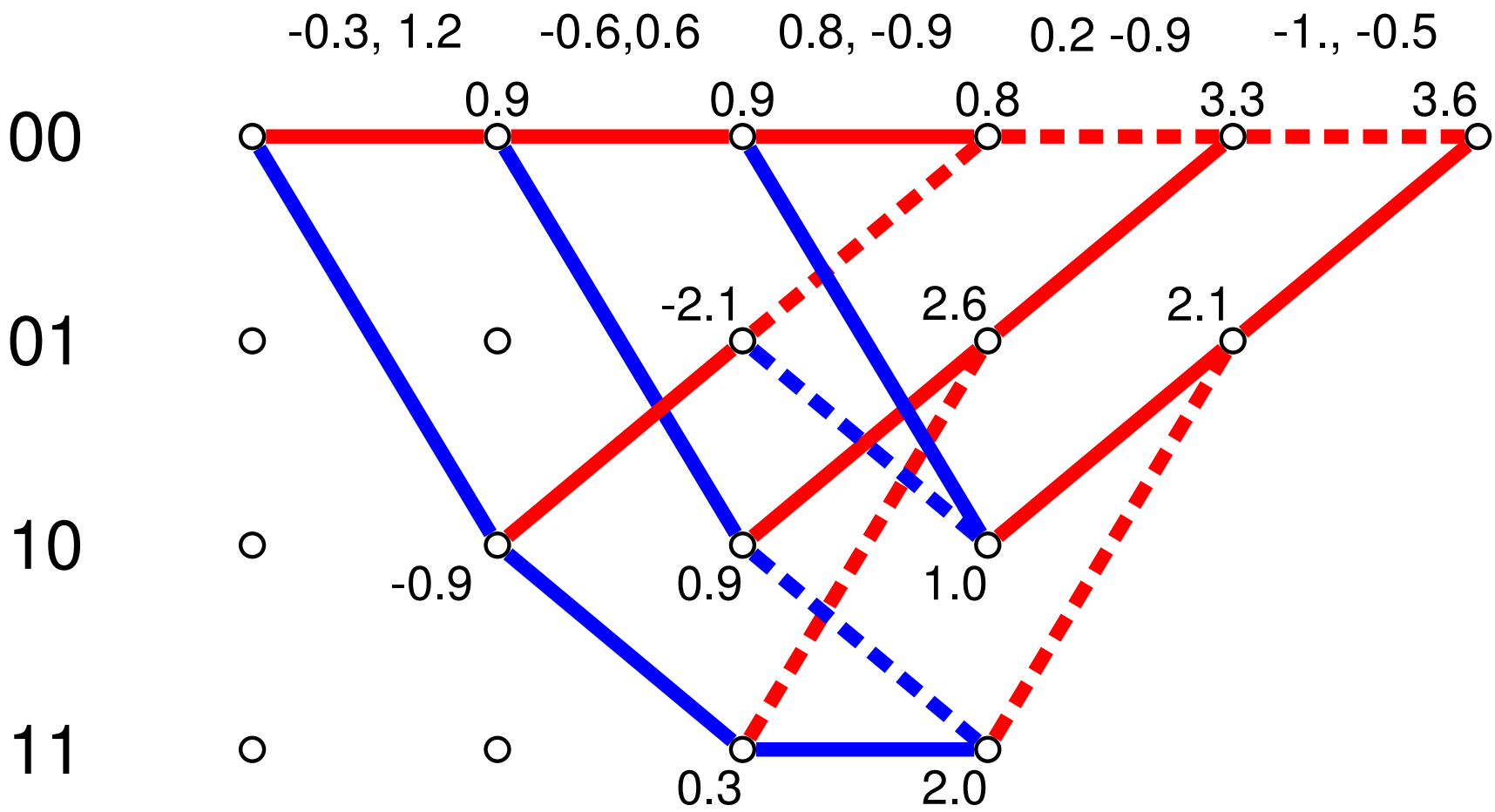
Truncated Trellis



Truncated Trellis



Truncated Trellis

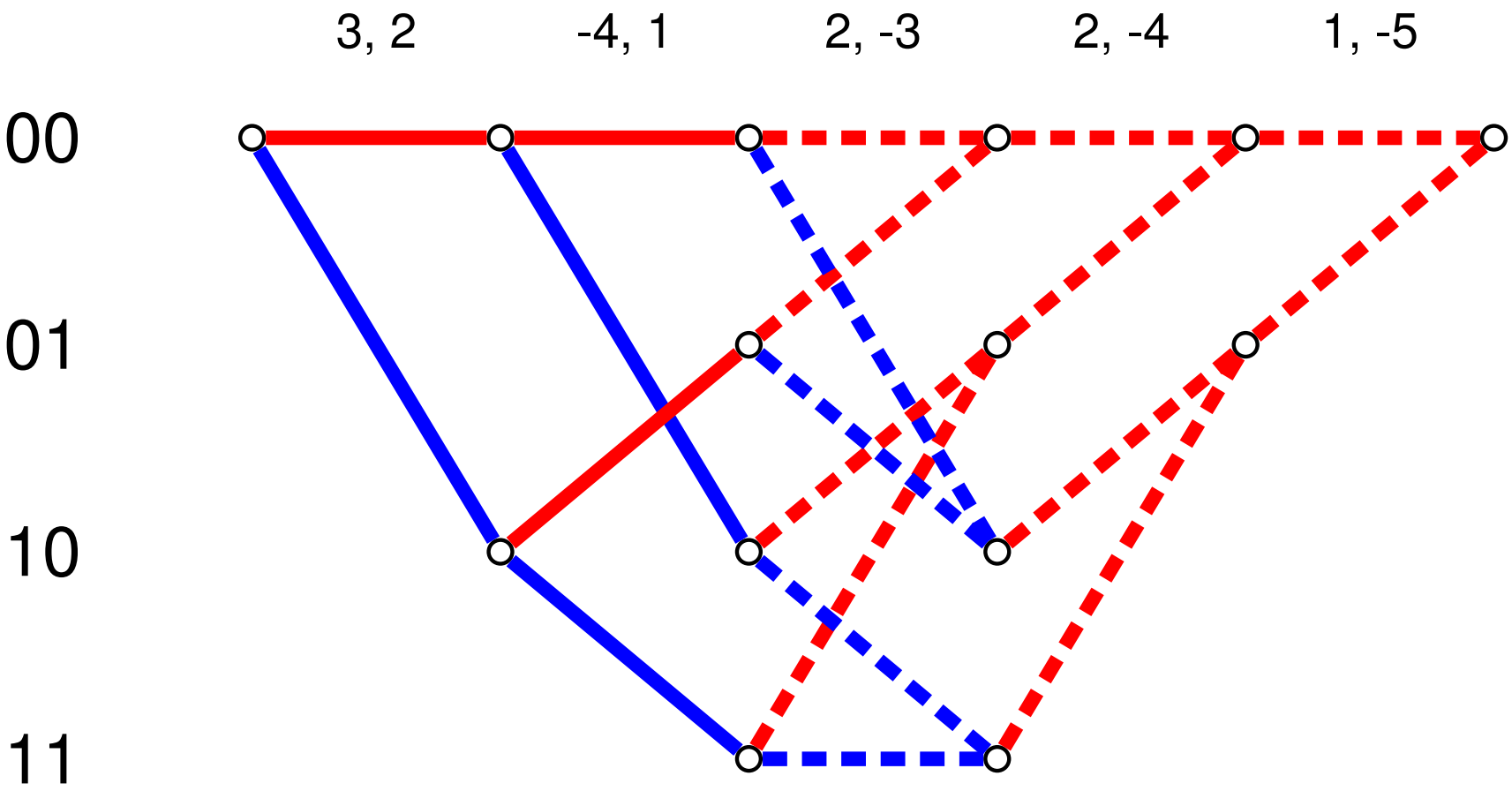


Truncated Trellis

$$
\begin{aligned}
r &= (-0.3,\ 1.2,\ -0.6,\ +0.6,\ +0.8,\ -0.9,\ 0.2,\ -0.9,\ -1.0,\ -0. \\
\hat{c} &= (+1.0,\ 1.0,\ +1.0,\ +1.0,\ -1.0,\ -1.0,\ 1.0,\ -1.0,\ -1.0,\ -1.
\end{aligned}
$$

$$
\begin{aligned}
D^2 &= ||r - \hat{c}|| \\
D^2 &= (1.3^2 + .2^2 + 1.6^2 + .4^2 + 1.8^2 + .1^2 + .8^2 + .1^2 + 0 + 0.5^2) \\
&= 8.6 \\
D &= 2.93
\end{aligned}
$$

Truncated Trellis



Dashed lines correspond to paths that are eliminated.

Weight Enumerator for Convolutional Codes

Suppose there are 3 information bits then 2 tail bits. The possible codewords are

Information Bits	Coded Bits	Weight	Bit Errors
000	0000000000	0	0
001	0000110111	5	1
010	0011011100	5	1
011	0011101011	6	2
100	1101110000	5	1
101	1101000111	6	2
110	1110101100	6	2
111	1110011011	7	3

There are 3 codewords of weight 5, 3 codewords of weight 6 and 1 codeword of weight 7.

$$A(x,y) = \sum_i \sum_j A_{i,j} x^i y^j$$

where $A_{i,j}$ is the number of codewords with i input ones and weight j.

Note that a convolutional code is a linear code. The bit error probability is upper bounded by

$$\begin{aligned} P_b &\leq \frac{1}{3}(3)P_2(5) + \frac{2}{3}(3)P_2(6) + \frac{3}{3}(1)P_2(7) \\ &= P_2(5) + 2P_2(6) + P_2(7) \end{aligned}$$

Weight Enumerator for Convolutional Codes

- The weight enumerator polynomial is a method for counting the number of codewords with
  - a given Hamming weight with
  - a given number of input ones and
  - a given certain length.
- It will be useful when determining bounds on the error probability.

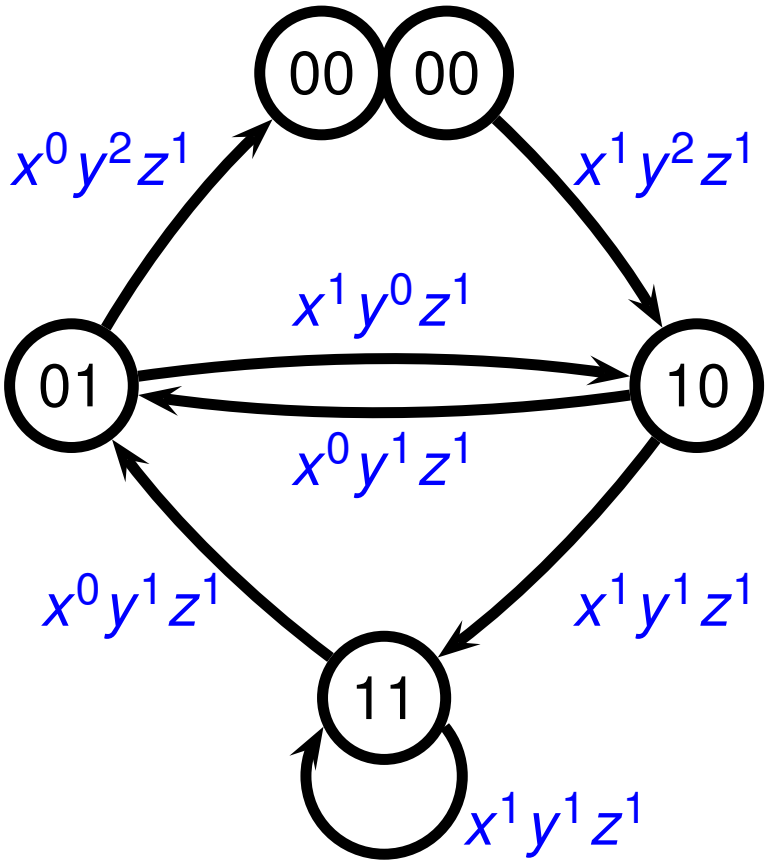
Weight Enumerator for 4 State Convolutional Code

- Consider the earlier example with four states.
- We would like to determine the number of paths that
  - start in the zero state,
  - diverge from the zero state for some time and
  - then remerge with the zero state such that
    - there are a certain number of input ones,
    - a certain number of output ones and
    - a certain length.

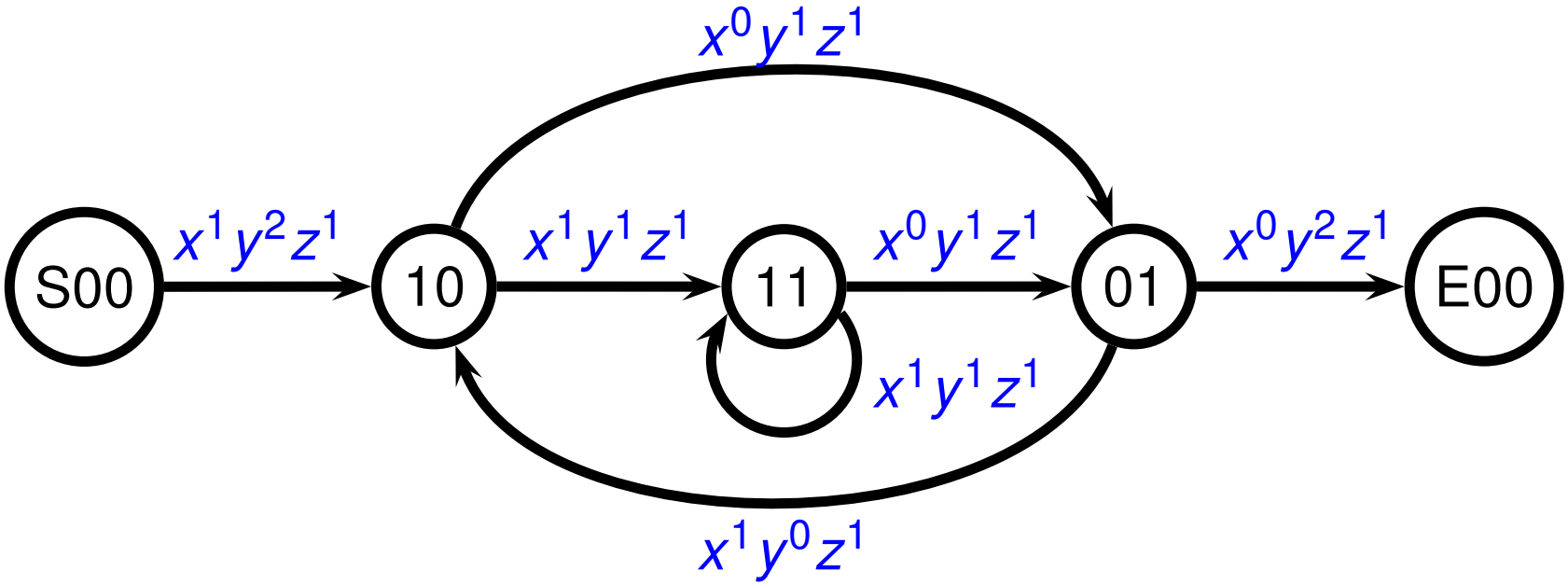
Weight Enumerator for 4 State Convolutional Code

- To do this let us split the zero state into a begining state and ending state.
- In addition we label each path with three variables (x, y, z).
- The power of x is the number of input ones;
- the power of y is the number of output ones;
- the power of z is the length of that path (namely one).
- To get the parameters for two consecutive paths we multiply these variables.

State Transition Diagram



Redrawn State Transition Diagram



Weight Enumerator

Let T_{10} represent the number of paths stating in the all zero state and ending in state 10. This includes paths that go through state 10 any number of times. Similarly let T_{11} be the number of paths stating from the 00 state and ending in state 11. Finally let T_{01} be the number of paths stating from the 00 state and ending in state 01. Then we can write the following equations

$$\begin{aligned} T_{10} &= xy^2z + xzT_{01} \\ T_{11} &= xyzT_{10} + xyzT_{11} \\ T_{01} &= yzT_{10} + yzT_{11} \end{aligned}$$

From these equations we can solve for T_{01}. We can rewrite these equations as

$$\begin{aligned} T_{01} - yzT_{10} - yzT_{11} &= 0 \\ -xzT_{01} + 1T_{10} &= xy^2z \\ -xyzT_{10} + (1 - xyz)T_{11} &= 0 \end{aligned}$$

Weight Enumerator

$$\begin{bmatrix} 1 & -yz & -yz \\ -xz & 1 & 0 \\ 0 & -xyz & (1-xyz) \end{bmatrix} \begin{bmatrix} T_{01} \\ T_{10} \\ T_{11} \end{bmatrix} = \begin{bmatrix} 0 \\ xy^2z \\ 0 \end{bmatrix}$$

Let

$$H = \begin{bmatrix} 1 & -yz & -yz \\ -xz & 1 & 0 \\ 0 & -xyz & (1-xyz) \end{bmatrix}$$

and

$$T = \begin{bmatrix} T_{01} \\ T_{10} \\ T_{11} \end{bmatrix}.$$

Weight Enumerator

Then

$$HT = \begin{bmatrix} 0 \\ xy^2z \\ 0 \end{bmatrix}.$$

$$H^{-1} = \begin{bmatrix} \frac{1-xyz}{1-xyz-xz^2y} & \frac{yz}{1-xyz-xz^2y} & \frac{yz}{1-xyz-xyz^2} \\ \frac{xz(1-xyz)}{1-xyz-xz^2y} & \frac{1-xyz}{1-xyz-xz^2y} & \frac{xyz^2}{1-xyz-xz^2y} \\ \frac{x^2yz^2}{1-xyz-xz^2y} & \frac{xyz}{1-xyz-xz^2y} & \frac{1-xyz^2}{1-xyz-xz^2y} \end{bmatrix}$$

Weight Enumerator

So

$$T = H^{-1}\begin{bmatrix} 0 \\ xy^2z \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{xy^3z^2}{1-xyz(1+z)} \\ \frac{xy^2z(1-xyz)}{1-xyz(1+z)} \\ \frac{x^2y^3z^2}{1-xyz(1+z)} \end{bmatrix}$$

Weight Enumerator

Then the number of paths stating at 00 and ending in 00 is $A(x, y, z) = y^2 z T_{01}$. In this case the solution is

$$\begin{aligned} A(x, y, z) &= \frac{x y^5 z^3}{1 - (xyz)(1 + z)} \\ &= x y^5 z^3 + x^2 y^6 z^4 + \cdots \end{aligned}$$

Thus there is one path through the trellis with one input one, 5 output ones and length 3. There is one path diverging and remerging with 2 input ones, length 4 and 6 output ones. The minimum (or free) distance of a convolutional code is the minimum number of output ones on any path that diverges from the all zero state and then remerges. This code has d_f of 5.

Weight Enumerator

We can extend this technique for deriving the transfer function by defining an adjacency matrix A where the (i,j) entry in the matrix is the monomial in x, y and z that represents the transition between state i and j. Consider then the adjacency matrix between all the nonzero states of a convolutional encoder. Then

$$T_i = \sum_j T_j A_{j,i} + G_i$$

where G_i is represents the transition from the all zero state to state i. Note that this is nonzero for only one value of i. The above equation leads to

$$\begin{aligned} [I - A]T &= G \\ T &= [I - A]^{-1} G \\ &= [H]^{-1} G \end{aligned}$$

If T_1 represents the state $000\cdots0001$. Then the transfer function from the all zero state and back is $T_1 A_{1,0}$.

Example

Consider the constraint length 3 code shown above. The number of paths stating at 00 and ending in 00 is $A(x, y, z) = y^2 z T_{01}$. In this case the solution is

$$\begin{aligned} A(x,y,z) &= \frac{xy^5z^3}{1-(xyz)(1+z)} \\ &= xy^5z^3 + x^2y^6z^4 + x^2y^6z^5 + x^3y^7z^5 \\ &\quad + 2x^3y^7z^6 + x^3y^7z^7 + x^4y^8z^6 + 3x^4y^8z^7 + \ldots \cdots \end{aligned}$$

Error Bounds for Convolutional Codes

- Thus there is one path through the trellis with one input one, 5 output ones and length 3.
- There is one path diverging and remerging with 2 input ones, length 4 and 6 output ones.
- The minimum (or free) distance of a convolutional code is the minimum number of output ones on any path that diverges from the all zero state and then remerges.
- This code has d_f of 5.

d_{min}

Error Bounds for Convolutional Codes

In order to compute the bit error probability of a convolutional code we need to compute another quantity, namely

$$\begin{aligned} w(y) &= \frac{\partial A(x,y,z)}{\partial x}\Big|_{x=1,z=1} \\ &= \sum_{l=d_{free}}^{\infty} w_l y^l \end{aligned}$$

where d_{free} is the minimum distance between any pair of codewords.

Error Bounds for Convolutional Codes

The bit error probability performance of convolutional codes can be upper bounded by

$$P_b \leq \sum_{d=d_{free}}^{\infty} w_d P_2(d) \leq \sum_{d=d_{free}}^{\infty} w_d D^d$$

where

- w_d is the average number of nonzero information bits on paths with Hamming distance d,
- $P_2(d)$ is the pairwise error probability between two codewords that differ in d places and D is a parameter that depends only on the channel.

Usually the summation in the upper bound is truncated to some finite number of terms.

Example 1

Binary symmetric channel (BSC) crossover probability p.

$$P_2(d) = \begin{cases} \sum_{l=t+1}^{d} \binom{d}{l} p^l (1-p)^{d-l} & d \text{ odd} \\ \sum_{l=t+1}^{d} \binom{d}{l} p^l (1-p)^{d-l} + \frac{1}{2}\binom{d}{d/2} p^{d/2}(1-p)^{d/2} & d \text{ even} \end{cases}$$

where $t = \lfloor (d-1)/2 \rfloor$.

$$P_2(d) \leq D^d$$

where

$$D = 2\sqrt{p(1-p)}$$

Example 2

Additive white Gaussian noise (AWGN) channel

$$P_2(d) = Q\left(\sqrt{\frac{2Ed}{N_0}}\right)$$

$$D = e^{-E/N_0}.$$

Performance Examples

- Generally hard decisions requires 2dB more signal energy than soft decisions for the same bit error probability.
- Also soft decisions is only about 0.25dB better than 8 level quantization.

Example Convolutional Code 1:

Constraint length 7, memory 6, 64 state decoder, rate 1/2 has the following upper bound.

$$P_b \leq 36D^{10} + 211D^{12} + 1404D^{14} + 11633D^{16} + \cdots$$

Example Convolutional Code 2:

Constraint length 9, memory 8, 256 state decoder, rate 1/2

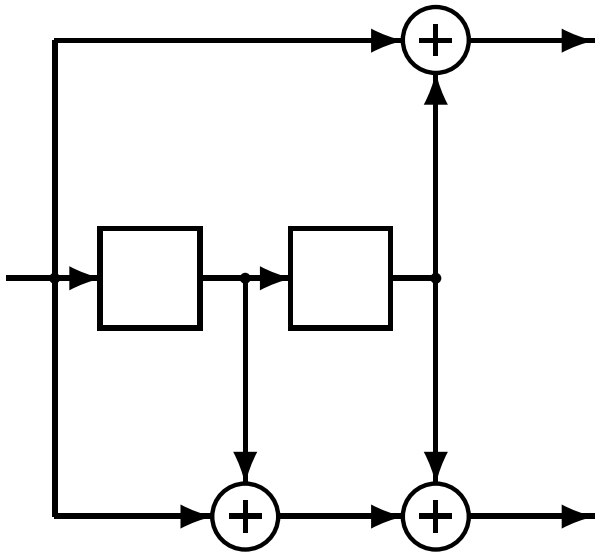
$$P_b \leq 33D^{12} + 281D^{14} + 2179D^{16} + 15035D^{18} + \cdots$$

Example Convolutional Code 3:

Constraint length 9, memory 8, 256 state decoder, rate 1/3

$$P_b \leq 11D^{18} + 32D^{20} + 195D^{22} + 564D^{24} + 1473D^{26} + \cdots$$

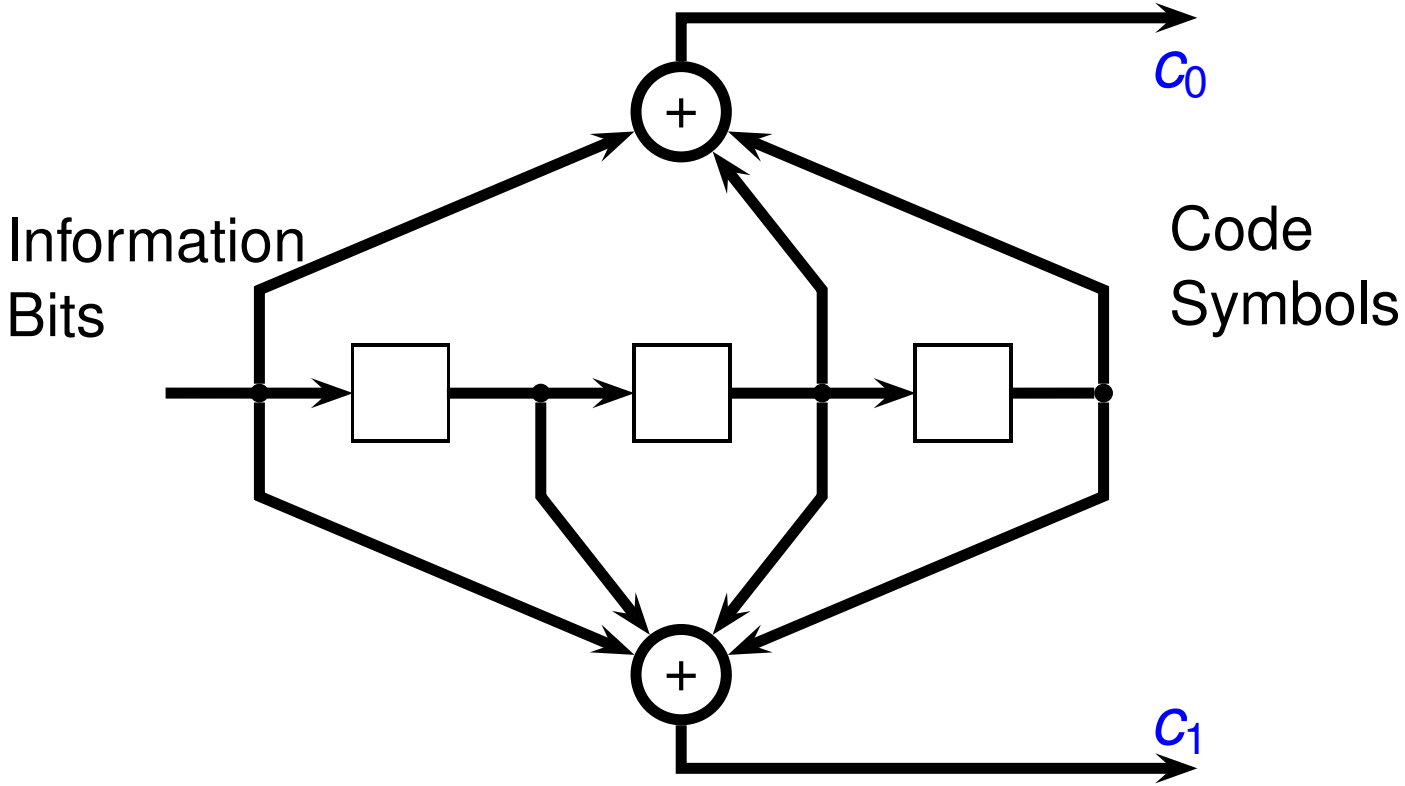
Example (K=3,M=2, rate 1/2 code)



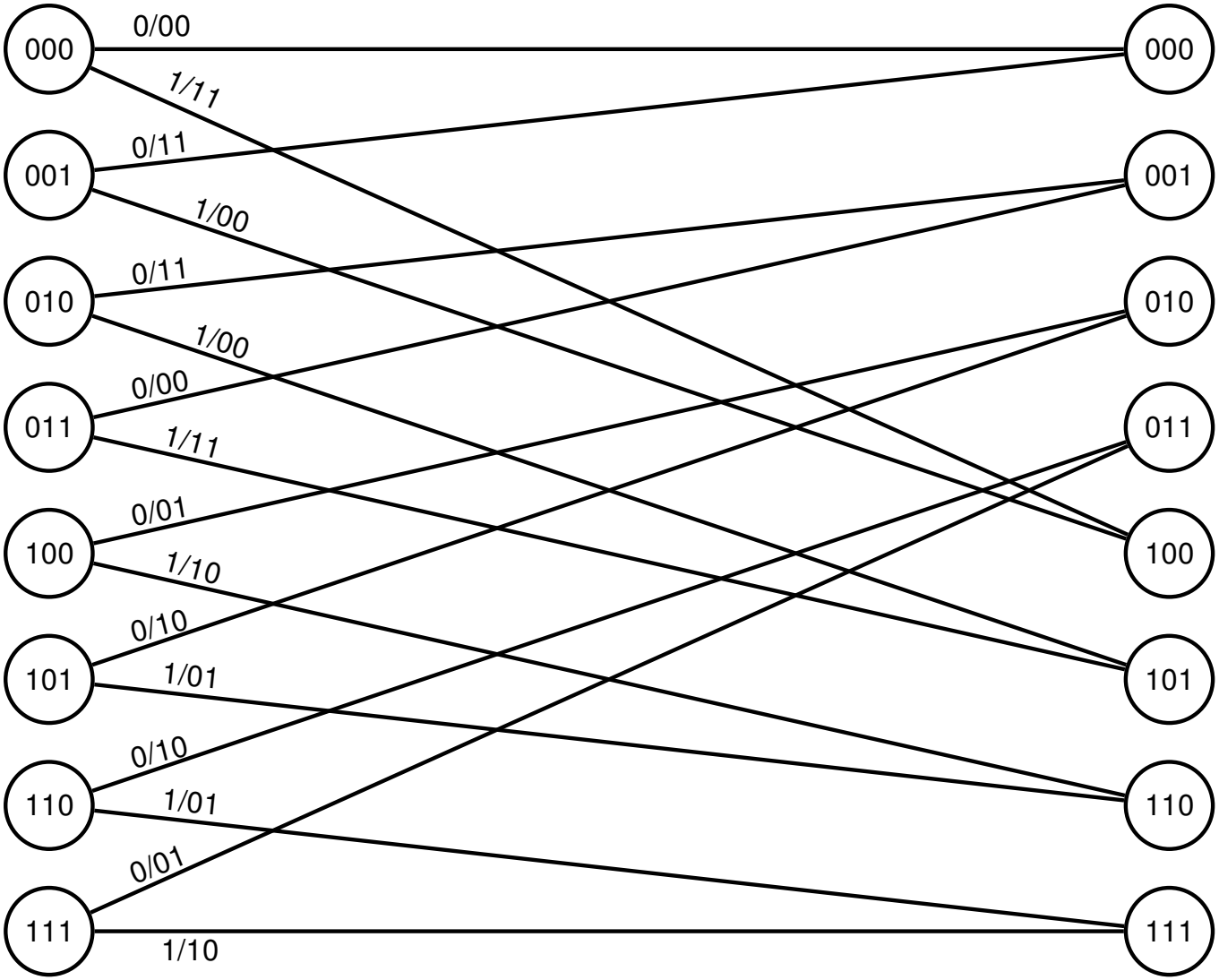
This code has d_f of 5. The weight enumerator polynomial is

$$w(D) = \frac{D^5}{(1-2D)^2} = D^5 + 4D^6 + 12D^7 + 32D^8 + \cdots$$

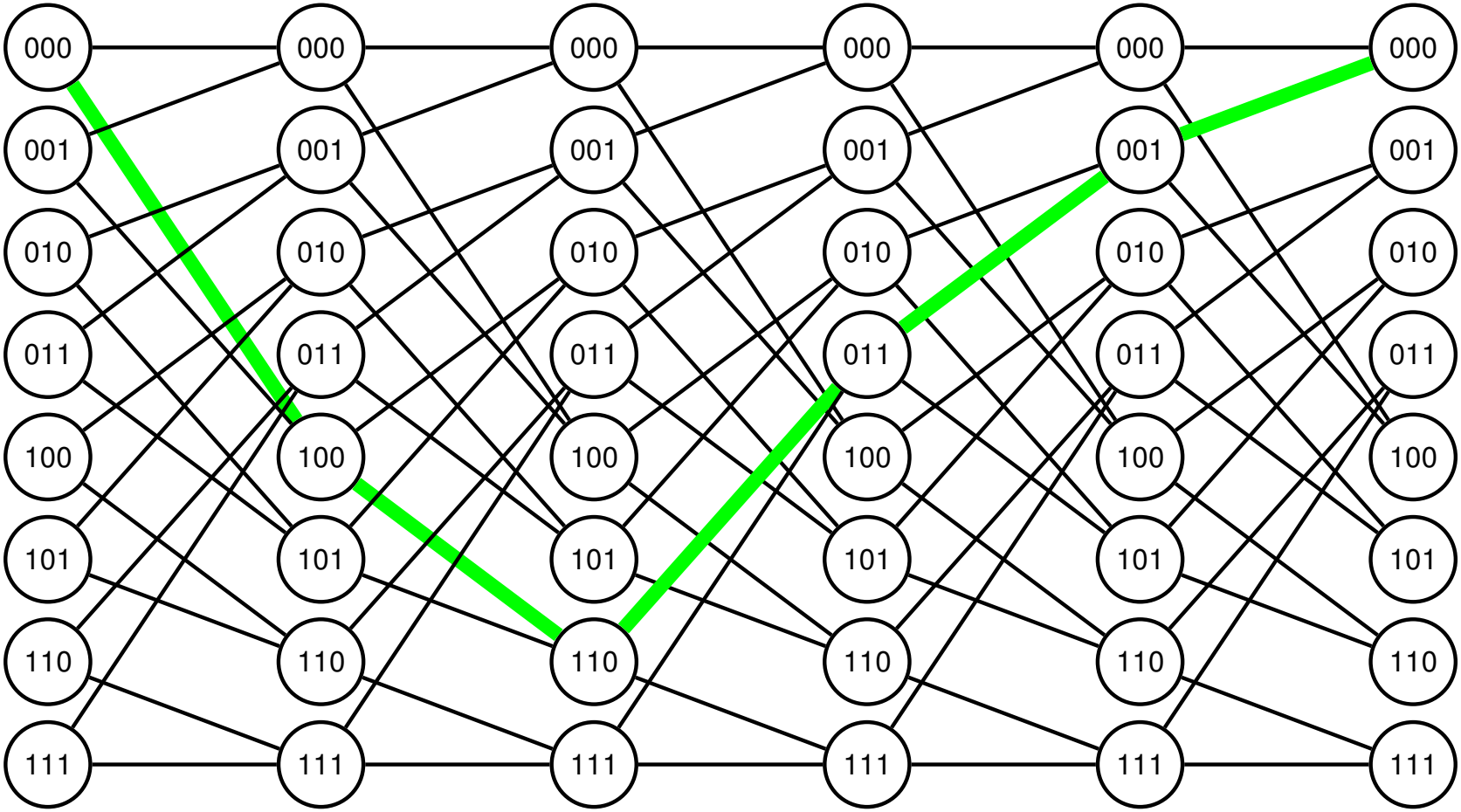
Example ($K = 4, M = 3$, rate 1/2 code): TI CC1101



Example ($K = 4$, $M = 3$, rate 1/2 code)



Example ($K = 4, M = 3$, rate 1/2 code)



Input=(1 1 0 0 0), Output= (11 10 10 00 11)

Example ($K = 4, M = 3$, rate 1/2 code)

$$\begin{aligned}
T_1 &= y^2 z T_2 + z T_3 \\
T_2 &= yz T_4 + yz T_5 \\
T_3 &= yz T_6 + yz T_7 \\
T_4 &= xy^2 z + xz T_1 \\
T_5 &= xz T_2 + xy^2 z T_3 \\
T_6 &= xyz T_4 + xyz T_5 \\
T_7 &= xyz T_6 + xyz T_7
\end{aligned}$$

Example ($K = 4, M = 3$, rate 1/2 code)

$$\begin{aligned}
-T_1 + y^2 z T_2 + z T_3 &= 0 \\
-T_2 + yz T_4 + yz T_5 &= 0 \\
-T_3 + yz T_6 + yz T_7 &= 0 \\
xz T_1 - T_4 &= -xy^2 z \\
xz T_2 + xy^2 z T_3 - T_5 &= 0 \\
xyz T_4 + xyz T_5 - T_6 &= 0 \\
xyz T_6 + (xyz - 1) T_7 &= 0
\end{aligned}$$

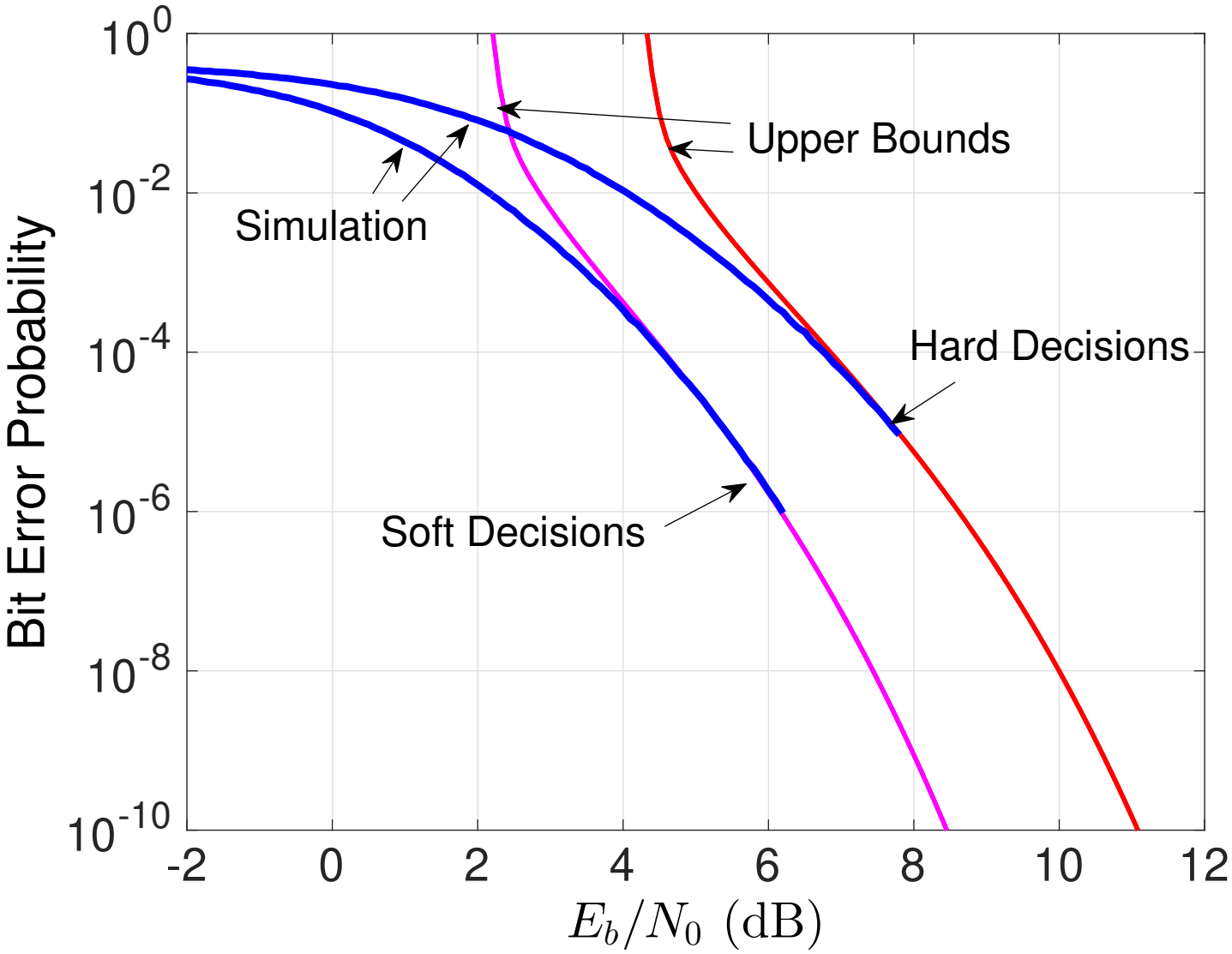
Example ($K = 4, M = 3$, rate 1/2 code)

$$\begin{bmatrix} -1 & y^2z & z & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & yz & yz & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & yz & yz \\ xz & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & xz & xy^2z & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & xyz & xyz & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & xyz & xyz-1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -xy^2z \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

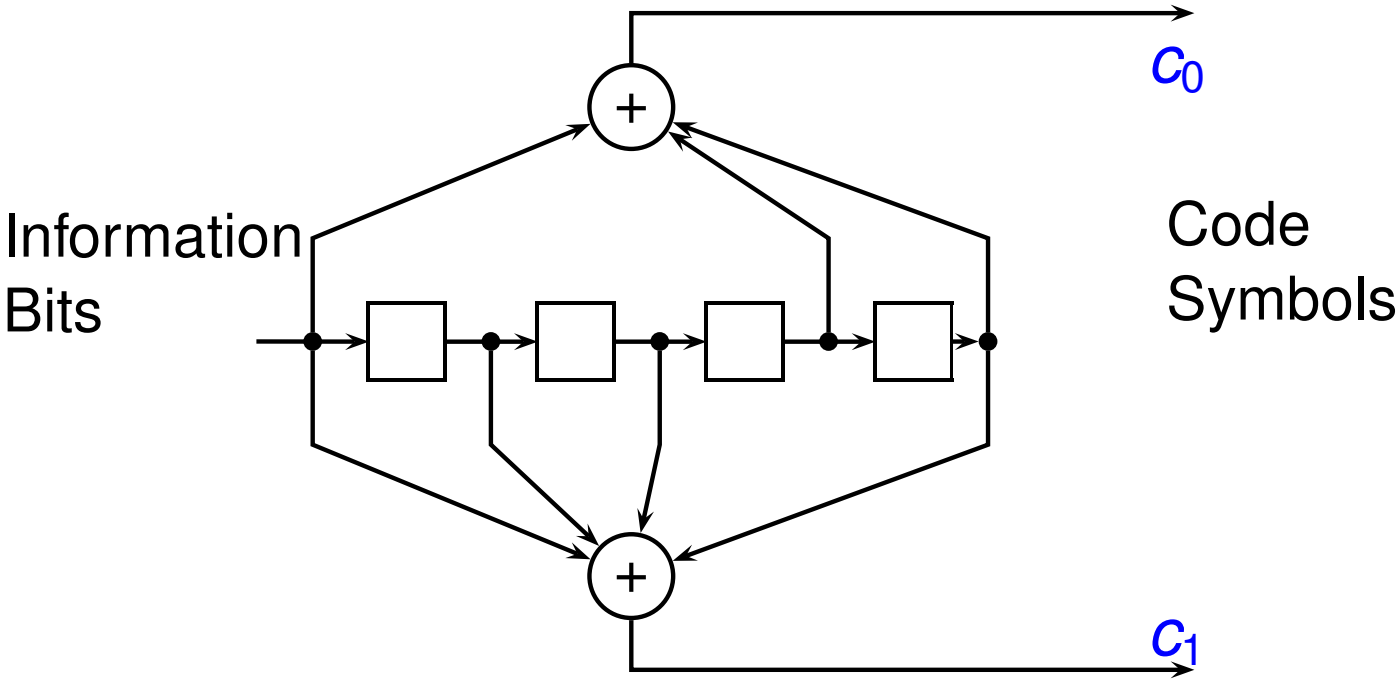
Example ($K = 4, M = 3$, rate 1/2 code)

$$
\begin{aligned}
T(x,y,z) &= \frac{y^2 z \left(x^2 y^4 z^4 + x y^5 z^3 - x^2 y^6 z^4\right)}{1 + x^2 y^4 z^4 - x^2 y^4 z^3 - x^2 y^2 z^4 + x^2 y^2 z^3 - x y^3 z^3 + -x y z^2 - x y z} \\
\frac{\partial T(x,y,z)}{\partial x}|_{x=1,z=1} &= \frac{y^2 \left(y^4 + y^5 - y^6\right)}{1 - 2y + y^3} \\
&= 2y^6 + 7y^7 + 18y^8 + 49y^9 + 130y^{10} + \ldots
\end{aligned}
$$

Example ($K = 4$, $M = 3$, rate 1/2 code)



Example ($K = 5, M = 4$, rate 1/2 code)

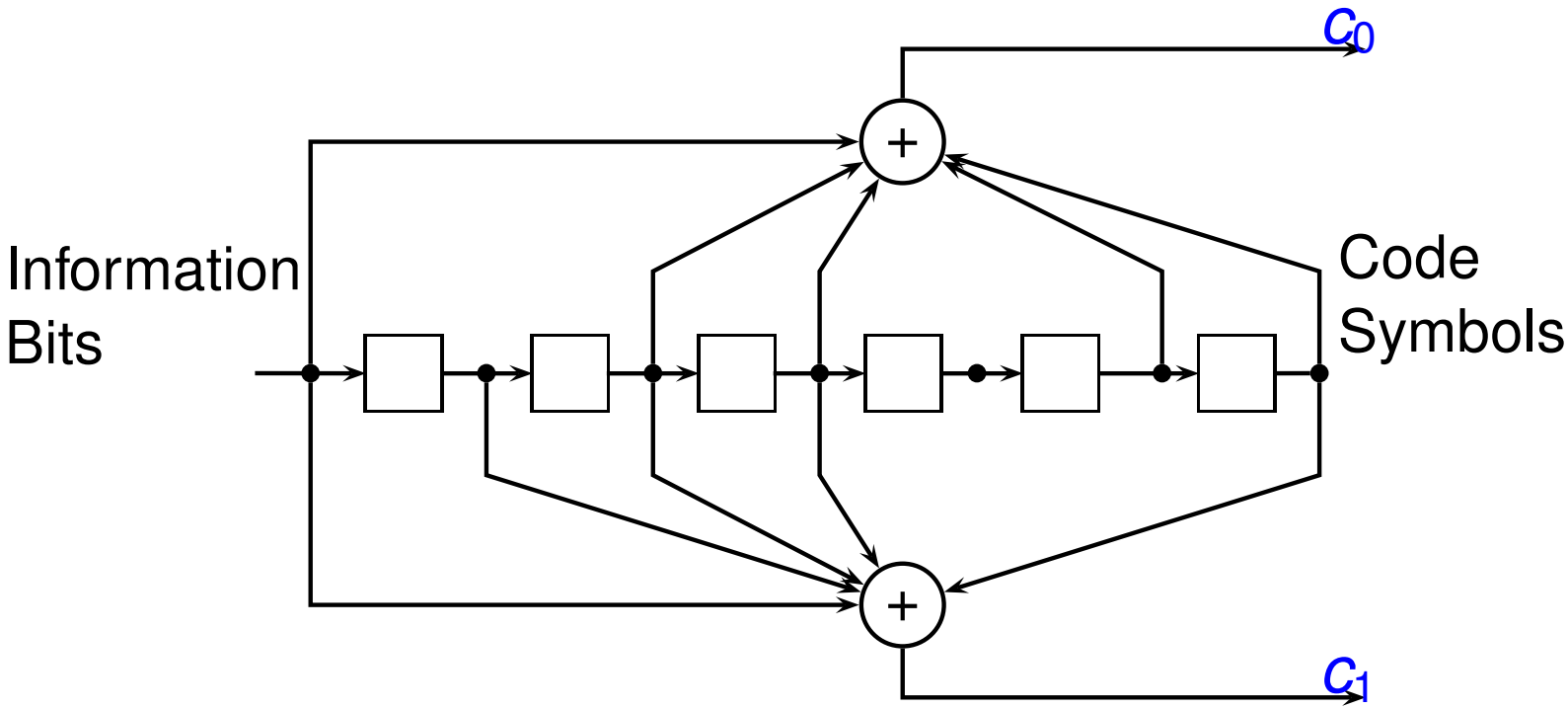


Example ($K = 5, M = 4$, rate 1/2 code)

$$\begin{aligned} w(y) &= y^7 a(y)/b(y) \\ a(y) &= 4 + 4y - 4y^3 - 11y^4 - 22y^5 + 5y^6 + 8y^7 + 27y^8 + 32y^9 - 10y^{10} \\ &\quad - 4y^{11} - 33y^{12} - 16y^{13} + 5y^{14} + 2y^{17} + 17y^{16} - 3y^{20} \\ b(y) &= \left(1 - y - 6y^3 - 2y^4 - 2y^5 + 3y^7 + 6y^8 + 4y^9 - 6y^{12} - 2y^{13} + 2y^{16}\right)^2 \end{aligned}$$

$$w(y) = 4y^7 + 12y^8 + 20y^9 + 72y^{10} + 225y^{11} + 500y^{12} + \cdots .$$

Example ($K = 7, M = 6$, rate 1/2 code)



Weight Enumerator for K=7, M=6 code

This code has d_f of 10. The weight enumerator polynomial for determining the bit error probability is given by

$$
\begin{aligned}
w(y) &= \frac{a(y)}{b(y)} \\
a(y) &= y^{10}\Big(36 - 77\,y^{2} - 140\,y^{4} + 813\,y^{6} + 269\,y^{8} - 4414\,y^{10} + 321\,y^{12} + 14884\,y^{14} - 5273\,y^{16} - 40509\,y^{18} \\
&\quad +39344\,y^{20} + 83884\,y^{22} - 177469\,y^{24} - 111029\,y^{26} + 608702\,y^{28} - 29527\,y^{30} - 1820723\,y^{32} + 817086\,y^{34} \\
&\quad +4951082\,y^{36} - 3436675\,y^{38} - 12279246\,y^{40} + 10300306\,y^{42} + 27735007\,y^{44} - 25648025\,y^{46} \\
&\quad -56773811\,y^{48} + 55659125\,y^{50} + 104376199\,y^{52} - 106695512\,y^{54} - 170819460\,y^{56} + 180836818\,y^{58} \\
&\quad +247565043\,y^{60} - 270555690\,y^{62} - 317381295\,y^{64} + 356994415\,y^{66} + 360595622\,y^{68} - 415401723\,y^{70} \\
&\quad -364292177\,y^{72} + 426295756\,y^{74} + 328382391\,y^{76} - 385686727\,y^{78} - 264812337\,y^{80} + 307287819\,y^{82} \\
&\quad +191225378\,y^{84} - 215144035\,y^{86} - 123515898\,y^{88} + 131946573\,y^{90} + 71124860\,y^{92} - 70570661\,y^{94} \\
&\quad -36310569\,y^{96} + 32722089\,y^{98} + 16308558\,y^{100} - 13052172\,y^{102} - 6380604\,y^{104} + 4433332\,y^{106} \\
&\quad +2147565\,y^{108} - 1265046\,y^{110} - 612040\,y^{112} + 297721\,y^{114} + 144665\,y^{116} - 56305\,y^{118} - 27569\,y^{120} \\
&\quad +8232\,y^{122} + 4066\,y^{124} - 874\,y^{126} - 435\,y^{128} + 60\,y^{130} + 30\,y^{132} - 2\,y^{134} - y^{136}\Big)
\end{aligned}
$$

Weight Enumerator for K=7, M=6 code

$$
\begin{aligned}
b(y) \quad = \quad & \Big(1 - 6\,y^4 - 4\,y^2 + 267\,y^{24} + 561\,y^{50} + 1184\,y^{42} - 403\,y^{20} - 1123\,y^{64} + 21746\,y^{44} - 1960\,y^{38} + 389\,y^{28} \\
& -1601\,y^{22} - 345\,y^{14} + 40\,y^8 + 85\,y^{10} + 844\,y^{18} - 16133\,y^{40} - 782\,y^{46} + 240\,y^{68} + 2509\,y^{26} - 2751\,y^{32} \\
& +2807\,y^{34} - 8766\,y^{56} - 32\,y^{72} + 3662\,y^{60} - 3064\,y^{30} + 2\,y^{76} \\
& -21403\,y^{48} + 3\,y^{66} + 131\,y^{58} + 15763\,y^{52} - 81\,y^{12} \\
& -331\,y^{54} + 262\,y^{16} - 30\,y^6 - 30\,y^{62} + 8344\,y^{36}\Big)^2
\end{aligned}
$$

$$
\begin{aligned}
w(D) \quad = \quad & 36D^{10} + 211D^{12} + 1404D^{14} + 11633D^{16} + 77433D^{18} \\
& + 502690D^{20} + 3322763D^{22} + 21292910D^{24} \\
& + 134365911D^{26} + 843425871D^{28} + \cdots
\end{aligned}
$$

Bit Error Probability Bounds

We can upper bound the bit error probability by

$$P_b \leq \sum_j w_j P_2(j) \leq \sum_j w_j D^j = w(D)$$

The first bound is the union bound. It is impossible to exactly evaluate this bound because there are an infinite number of terms in the summation. Dropping all but the first N terms gives an approximation. It may no longer be an upper bound though. If the weight enumerator is known we can get arbitrarily close to the union bound and still get a bound as follows.

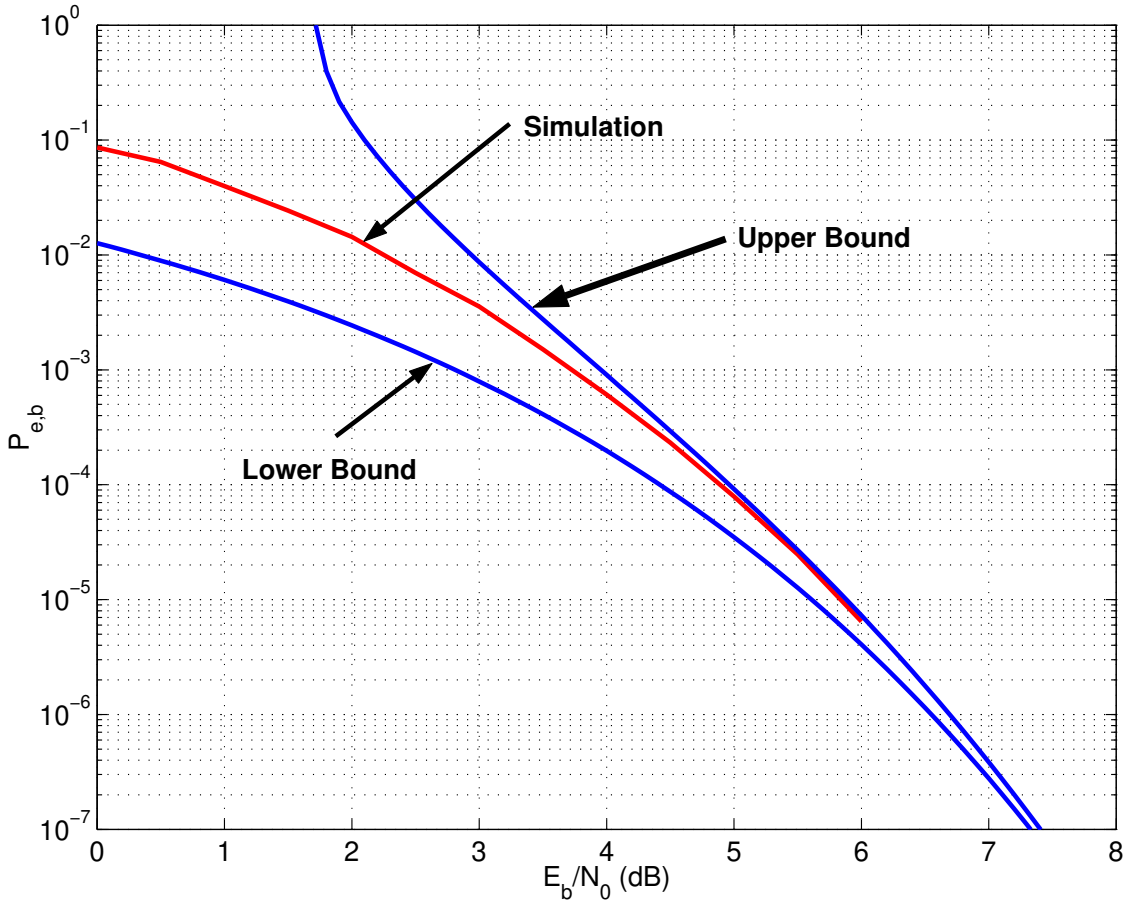
Bit Error Probability Bounds

$$
\begin{aligned}
P_b &\leq \sum_j w_j P_2(j) = \sum_{j=d_f}^{N} w_j P_2(j) + \sum_{j=N+1}^{\infty} w_j P_2(j) \\
&\leq \sum_{j=d_f}^{N} w_j P_2(j) + \sum_{j=N+1}^{\infty} w_j D^j \\
&= \sum_{j=d_f}^{N} w_j (P_2(j) - D^j) + \sum_{j=d_f}^{\infty} w_j D^j = \sum_{j=d_f}^{N} w_j (P_2(j) - D^j) + w(D)
\end{aligned}
$$

Bit Error Probability Bounds

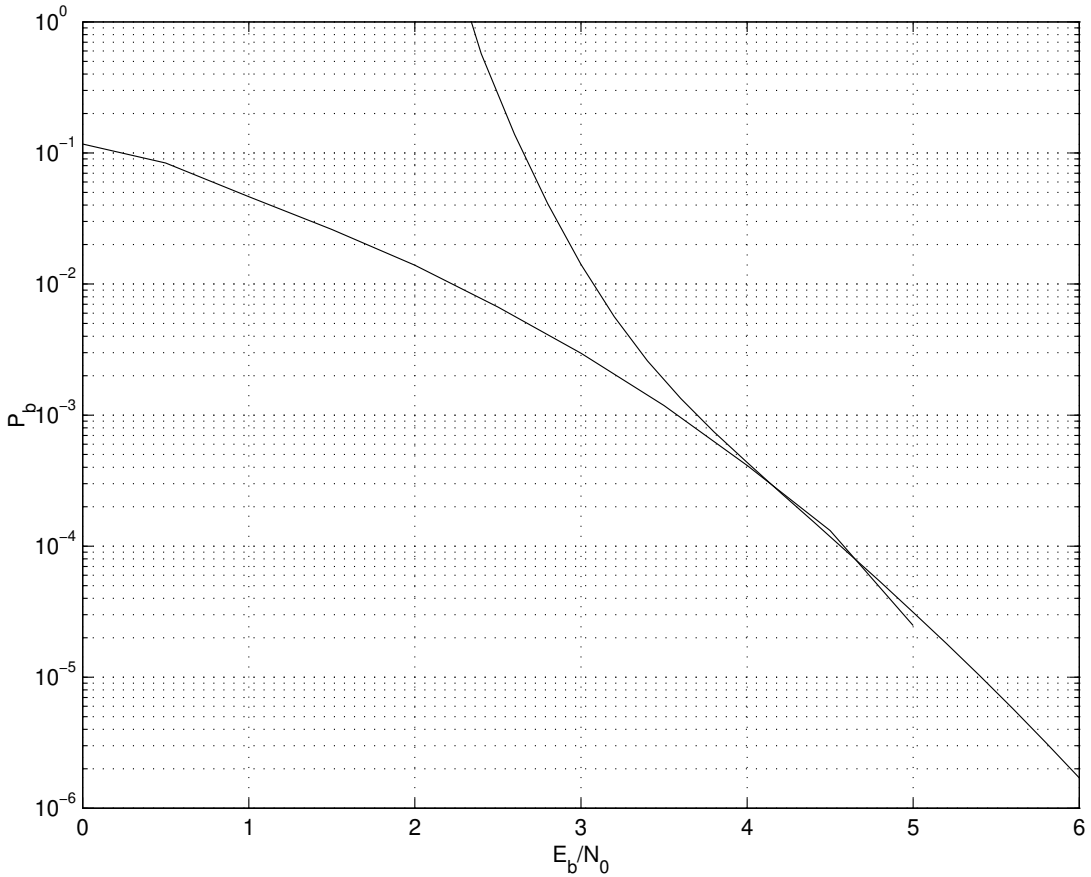
- The second term is the Union-Bhattacharyya (U-B) bound.
- The first term is clearly less than zero, so we get something that is tighter than the U-B bound.
- By choosing N sufficiently large we can sometimes get significant improvements over the U-B bound.

Bit Error Probability Bounds



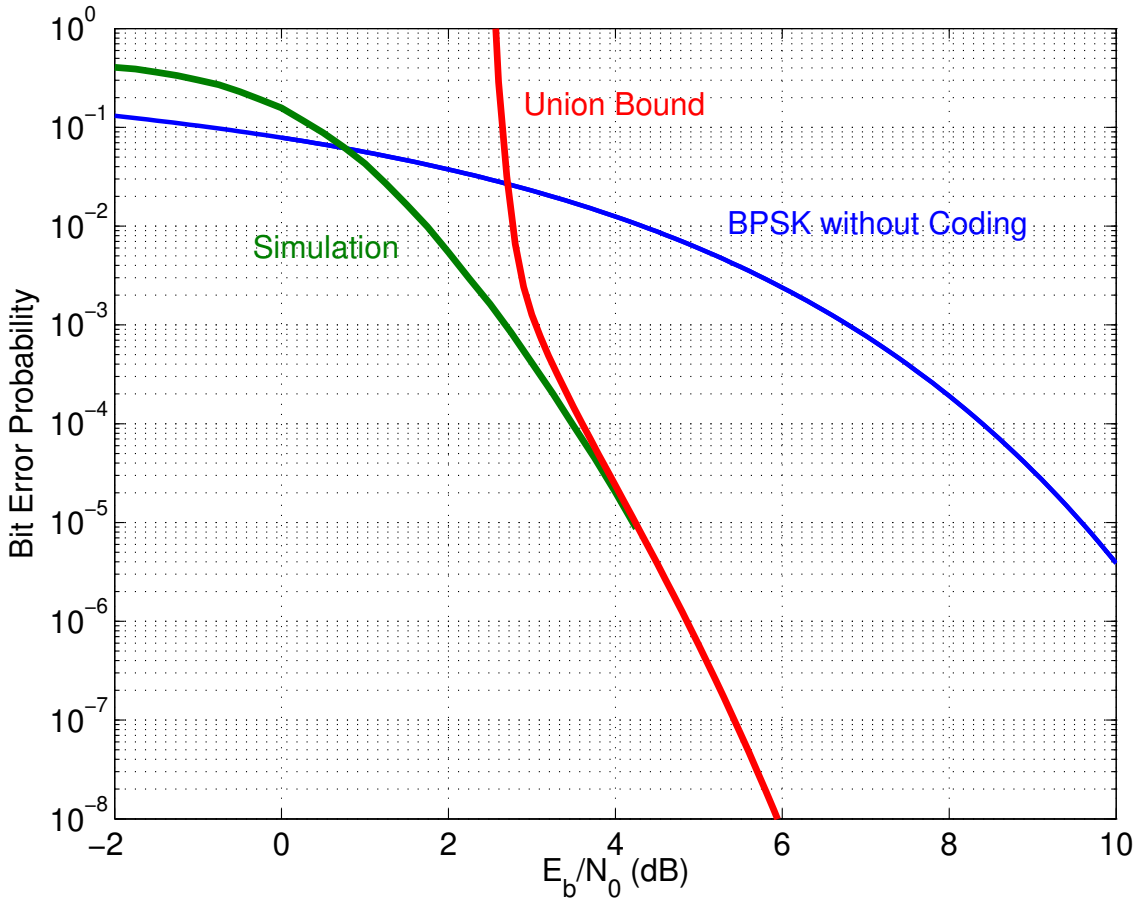
Error probability of constraint length 3 convolutional codes on an additive white Gaussian noise channel with soft decisions decoding (upperbound, simulation and lower bound).

Bit Error Probability for $K = 4$ Code

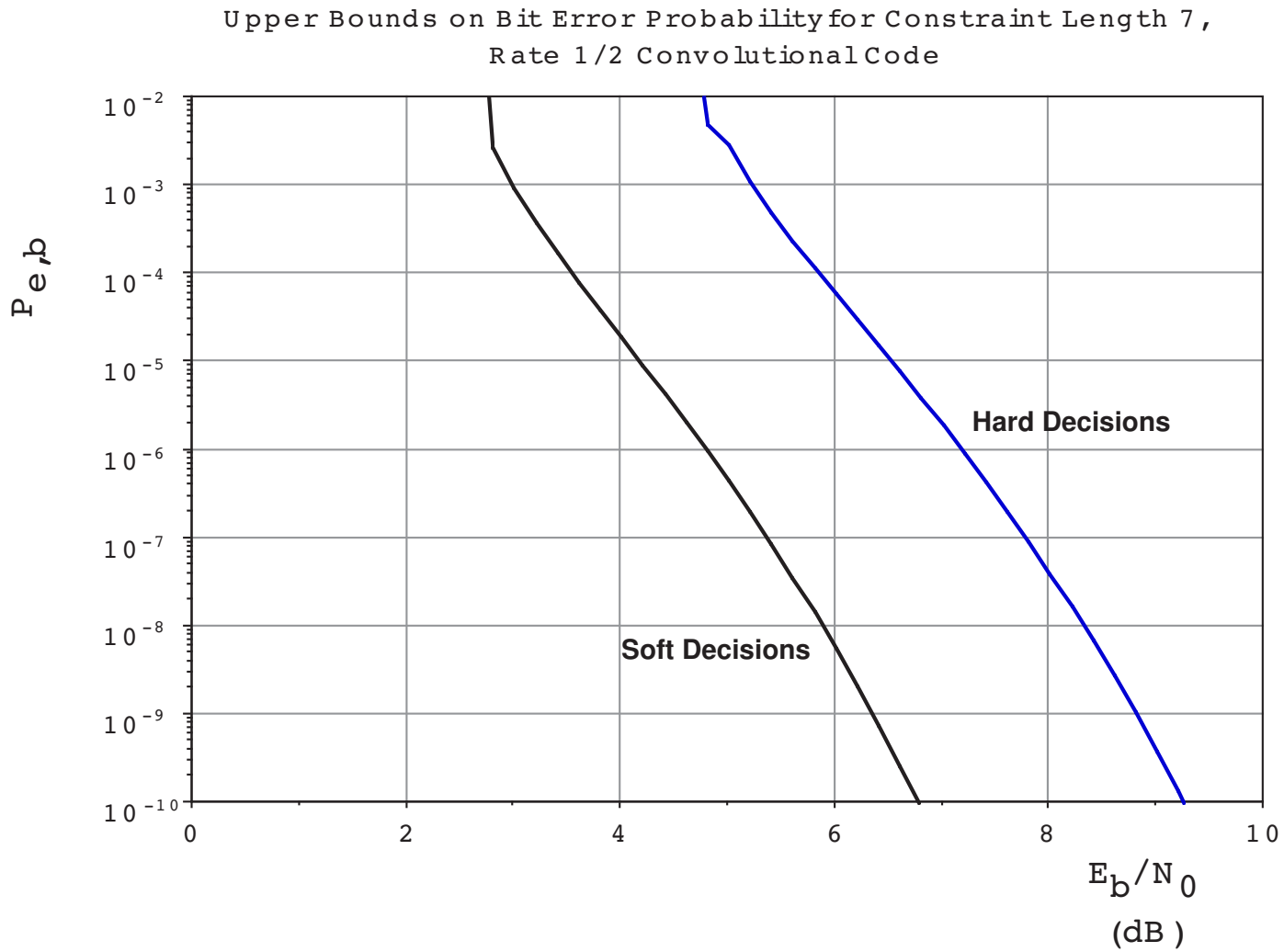


Error probability of constraint length 4 convolutional codes on an additive white Gaussian noise channel with soft decisions decoding (upperbound, simulation).

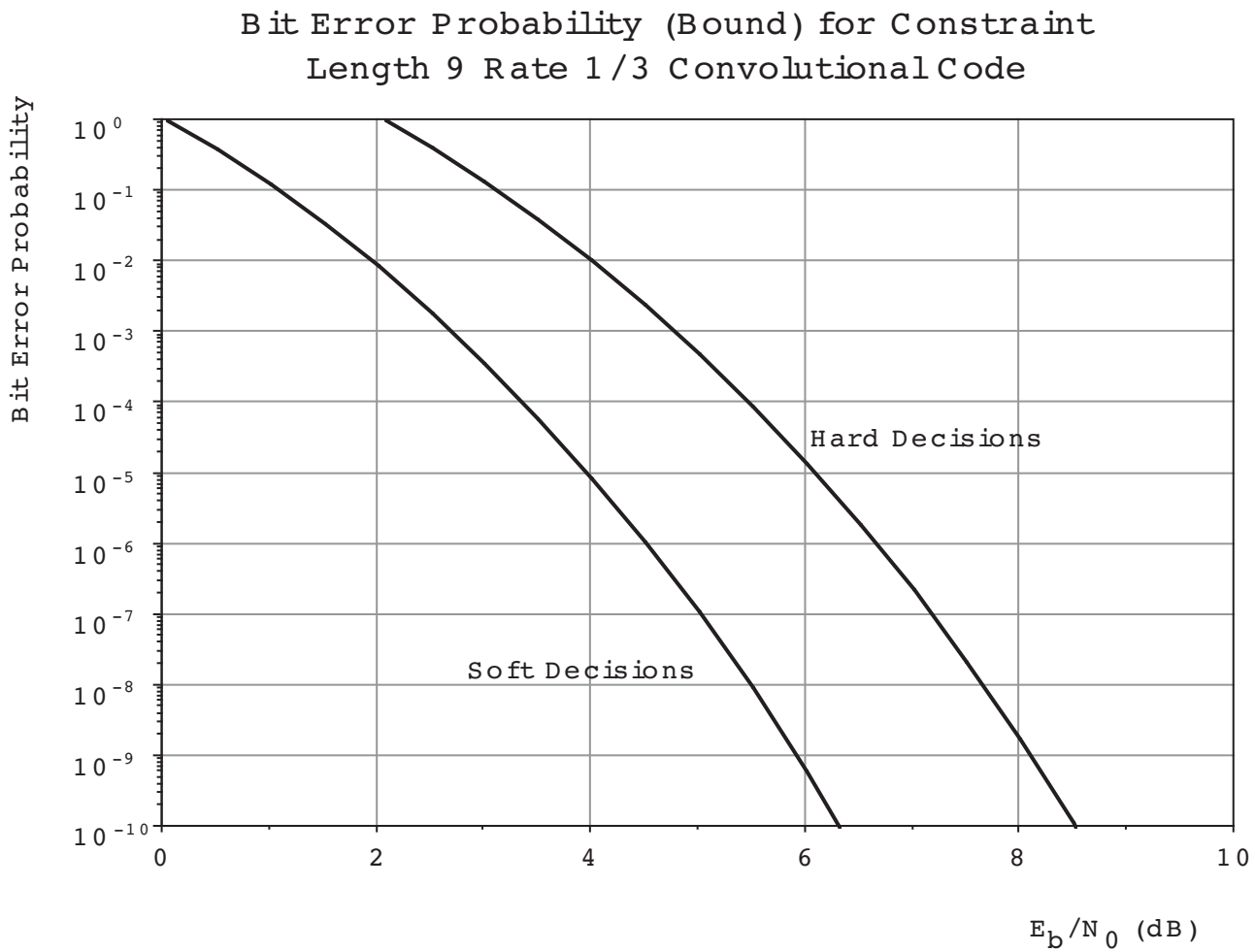
Bit Error Probability for $K = 7$ Code



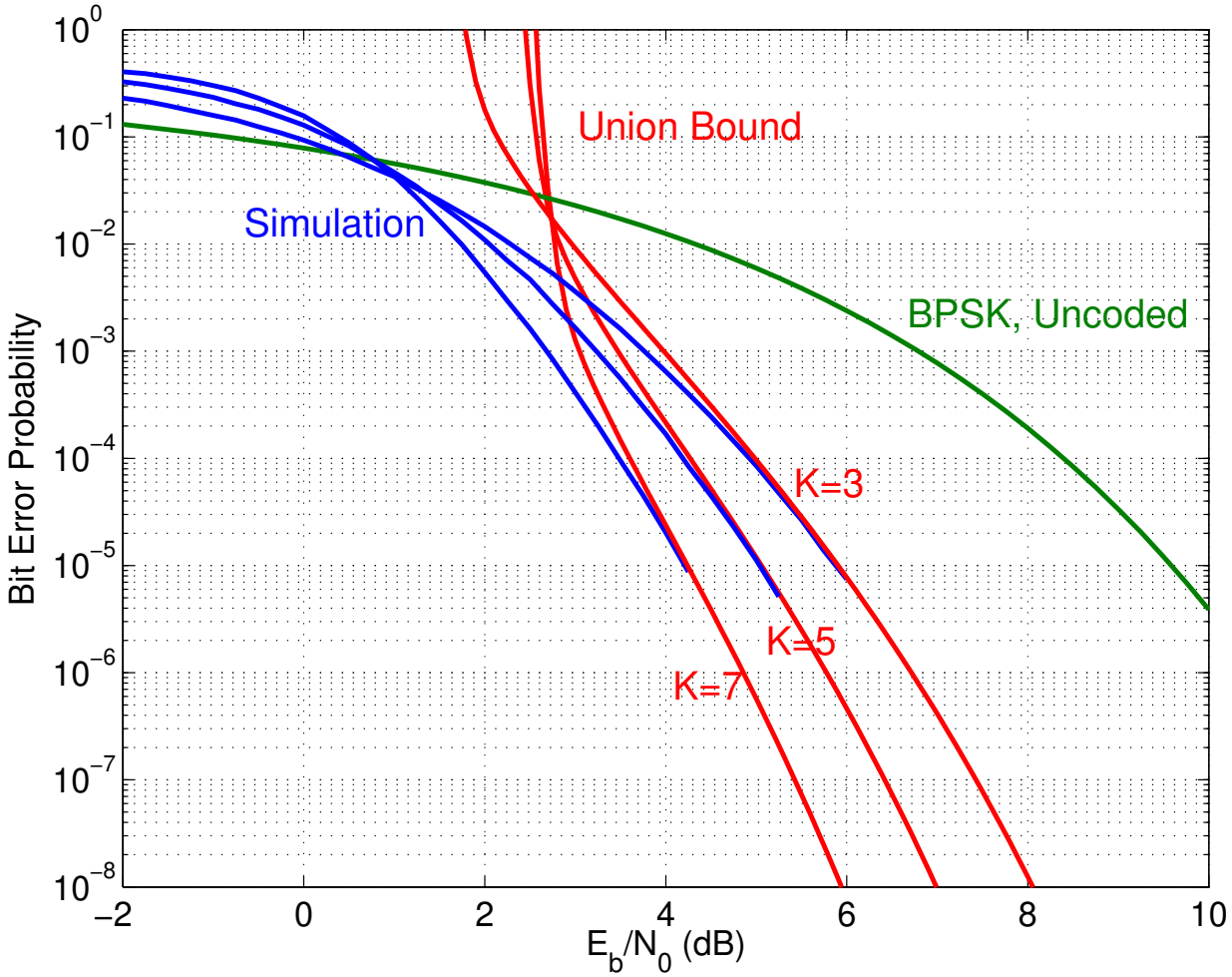
Bit Error Probability for $K = 7$ Code: Soft Decisions, Hard Decisions



Bit Error Probability for $K = 9$ Code: Soft Decisions, Hard Decisions



Bit Error Rate of Convolutional Codes

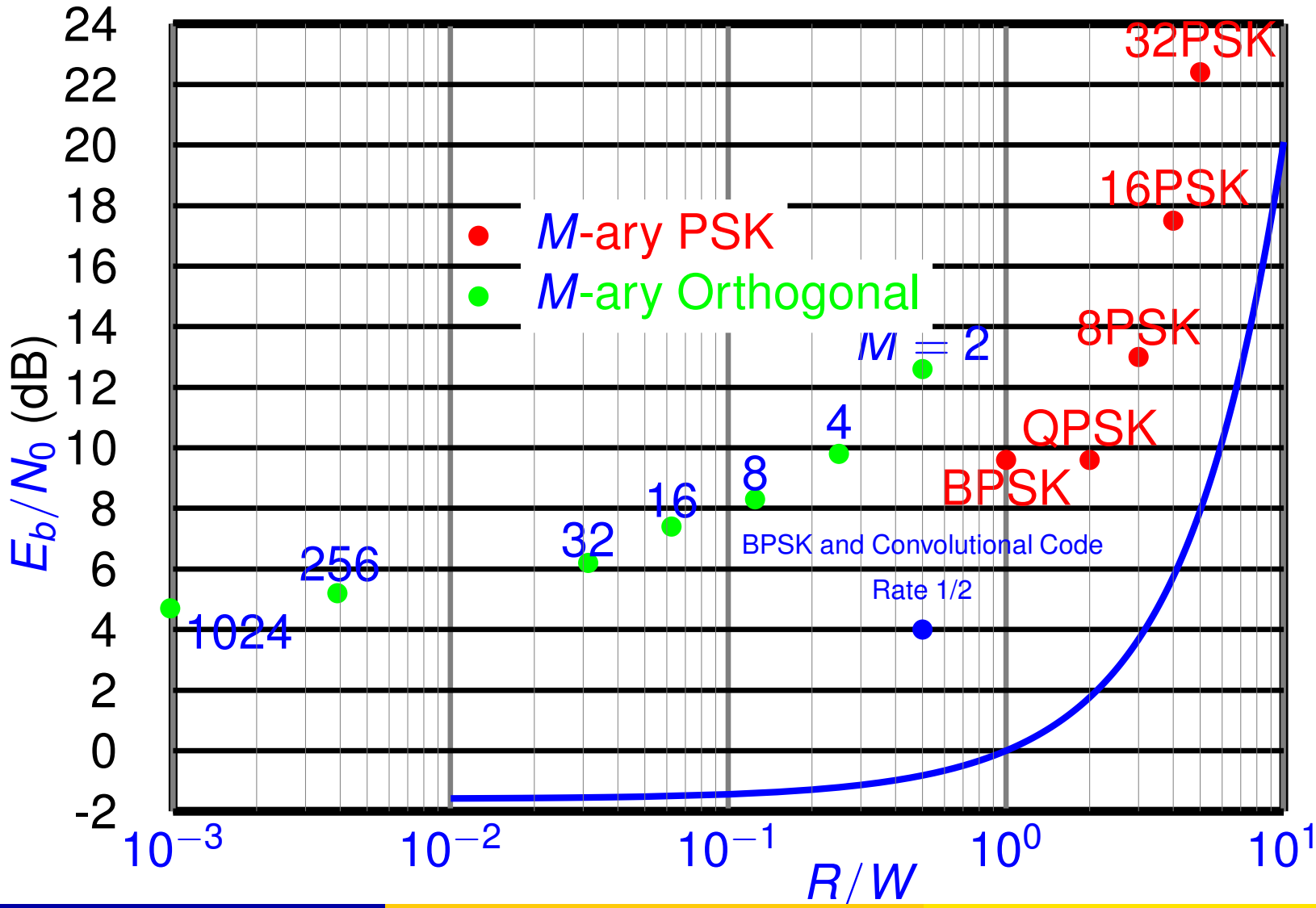


Convolutional Decoder $p = 0.05$

Convolutional Decoder $p = 0.1$

Convolutional Decoder $p = 0.2$

Capacity


24
22
20
18
16
14
12
10
8
6
4
2
0
-2
E_b/N_0 (dB)
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
R/W
32PSK
16PSK
8PSK
QPSK
BPSK
M-ary PSK
M-ary Orthogonal
$M = 2$
4
8
16
32
256
1024
BPSK and Convolutional Code
Rate 1/2