16.548 Notes 15:

Concatenated Codes, Turbo Codes and Iterative Processing

Outline

- Introduction
  - Pushing the Bounds on Channel Capacity
  - Theory of Iterative Decoding
  - Recursive Convolutional Coding
  - Theory of Concatenated codes
- Turbo codes
  - Encoding
  - Decoding
  - Performance analysis
  - Applications
- Other applications of iterative processing
  - Joint equalization/FEC
  - Joint multiuser detection/FEC

Shannon Capacity Theorem

$$C = \frac{1}{2}\log_2\left(1 + \frac{2R_c E_b}{N_0}\right)$$

$$R_c \le \frac{C}{1 - H_b(e)}$$

From the converse to the coding theorem

Find (P_b, E_b/N_0) pair $\Longrightarrow$ Limit for given code rate

Capacity as a function of Code rate

Example: Binary-input (+/- A), AWGN channel

$$C = \frac{1}{2}\int_{-\infty}^{\infty} p(y \mid A)\log_2 \frac{p(y \mid A)}{p(y)}dy + \frac{1}{2}\int_{-\infty}^{\infty} p(y \mid -A)\log_2 \frac{p(y \mid -A)}{p(y)}dy$$

Motivation: Performance of Turbo Codes.

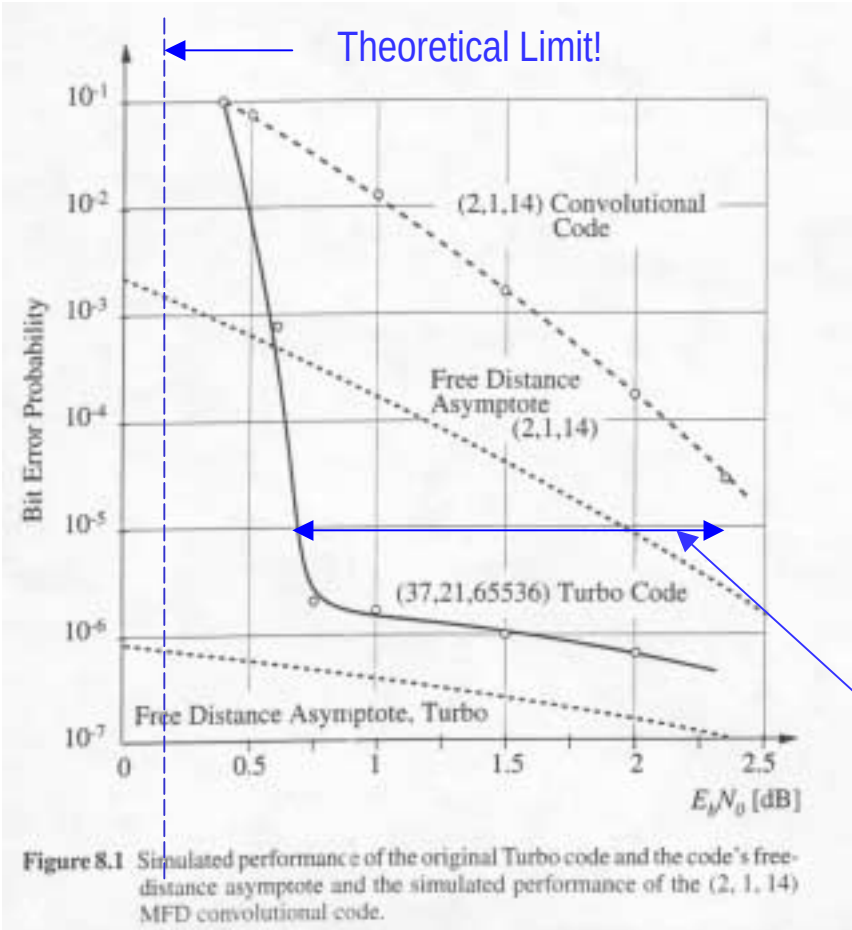


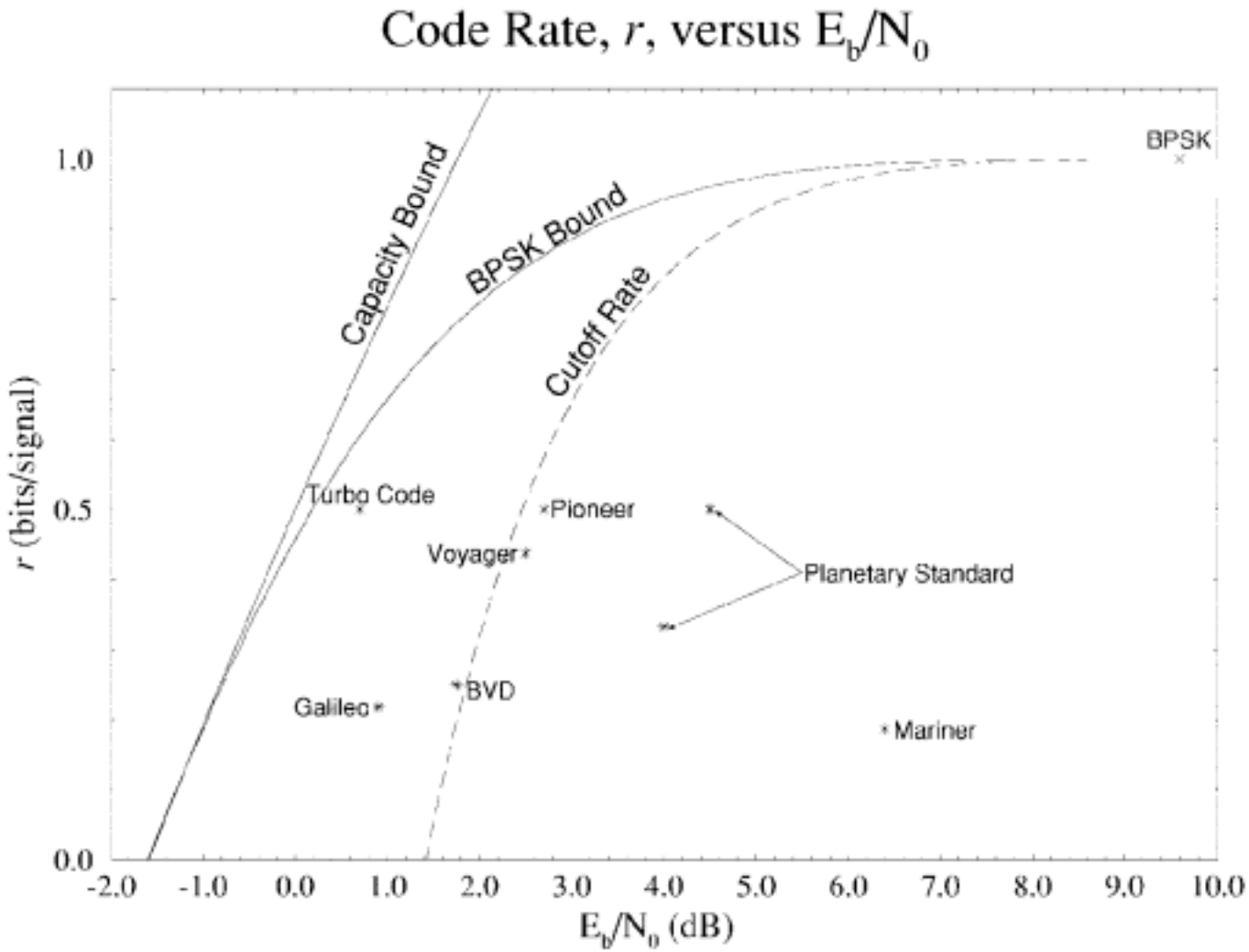
Figure 8.1 Simulated performance of the original Turbo code and the code's free-distance asymptote and the simulated performance of the (2, 1, 14) MFD convolutional code.

- Comparison:
  - Rate 1/2 Codes.
  - K=5 turbo code.
  - K=14 convolutional code.
- Plot is from:

  L. Perez, "Turbo Codes", chapter 8 of Trellis Coding by C. Schlegel. IEEE Press, 1997.

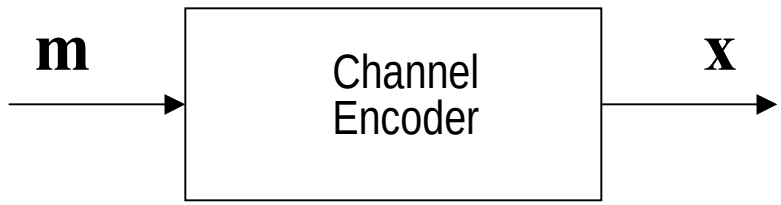
Gain of almost 2 dB!

Power Efficiency of Existing Standards



Error Correction Coding

- Channel coding adds structured redundancy to a transmission.



  - The input ***message*** **m** is composed of K symbols.
  - The output ***code word*** **x** is composed of N symbols.
  - Since N > K there is redundancy in the output.
  - The ***code rate*** is r = K/N.
- Coding can be used to:
  - Detect errors: **ARQ**
  - Correct errors: **FEC**

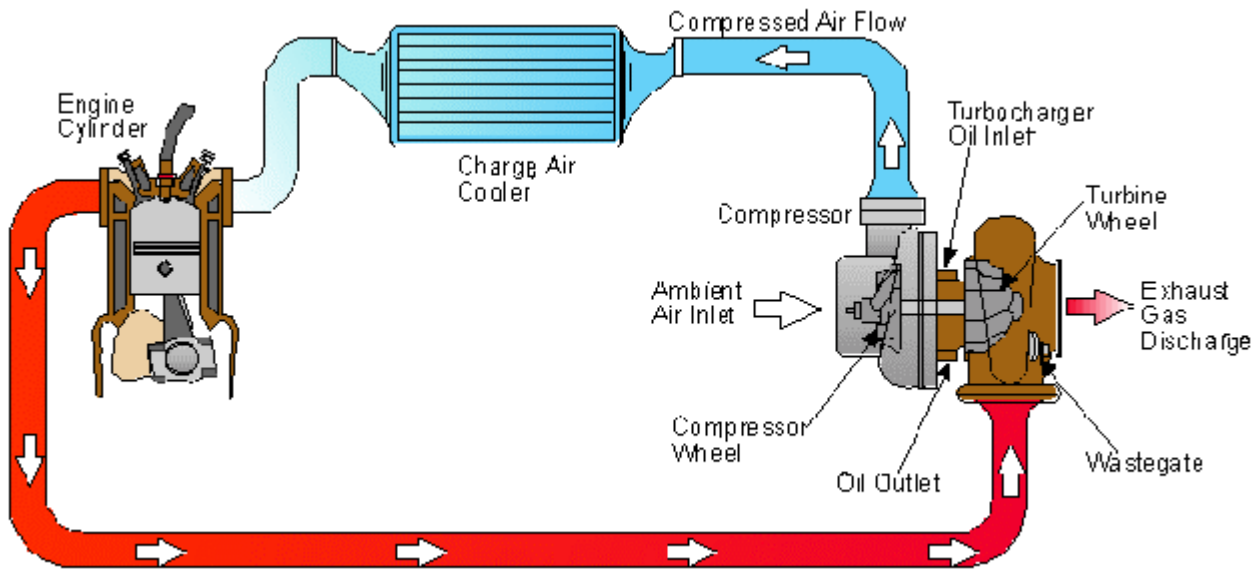
Traditional Coding Techniques

- Block Codes
  - Long Codes
  - Hard desisjon
  - Good at low bit error rates (10^{-4} and down)
  - Many types
    - Ex. Hamming, Golay, Reed-Muller, Reed Solomon, BCH, etc
  - Used for instance in CD players (RS), Intelsat (RS), Voyager (Golay)

- Convolutional Codes
  - Short Codes
  - Soft desisjon (Viterbi)
  - Good at high bit-error rates (10^{-3} - 10^{-4})
  - Used in Inmarsat, 3G and various wireless systems
  - Long constraint length codes with sequential decoding

The Turbo-Principle/Iterative Decoding

- Turbo codes get their name because the decoder uses feedback, like a turbo engine.



Theory of Iterative Coding

$$L(d|x) = \log\left[\frac{P(d=+1|x)}{P(d=-1|x)}\right] = \log\left[\frac{p(x|d=+1)\ P(d=+1)}{p(x|d=-1)\ P(d=-1)}\right] \quad (6)$$

$$L(d|x) = \log\left[\frac{p(x|d=+1)}{p(x|d=-1)}\right] + \log\left[\frac{P(d=+1)}{P(d=-1)}\right] \quad (7)$$

$$L(d|x) = L(x|d) + L(d) \quad (8)$$

where $L(x|d)$ is the LLR of the test statistic x obtained by measurements of the channel output x under the alternate conditions that $d = +1$ or $d = -1$ may have been transmitted, and $L(d)$ is the a priori LLR of the data bit d.

To simplify the notation, Equation (8) is rewritten as follows:

$$L'(\hat{d}) = L_c(x) + L(d) \quad (9)$$

where the notation $L_c(x)$ emphasizes that this LLR term is the result of a channel measurement made at the receiver.

Theory of Iterative Coding(2)

For a systematic code, it can be shown [3] that the LLR (soft output) $L(\hat{d})$ out of the decoder is equal to Equation 10:

$$L(\hat{d}) = L'(\hat{d}) + L_e(\hat{d}) \tag{10}$$

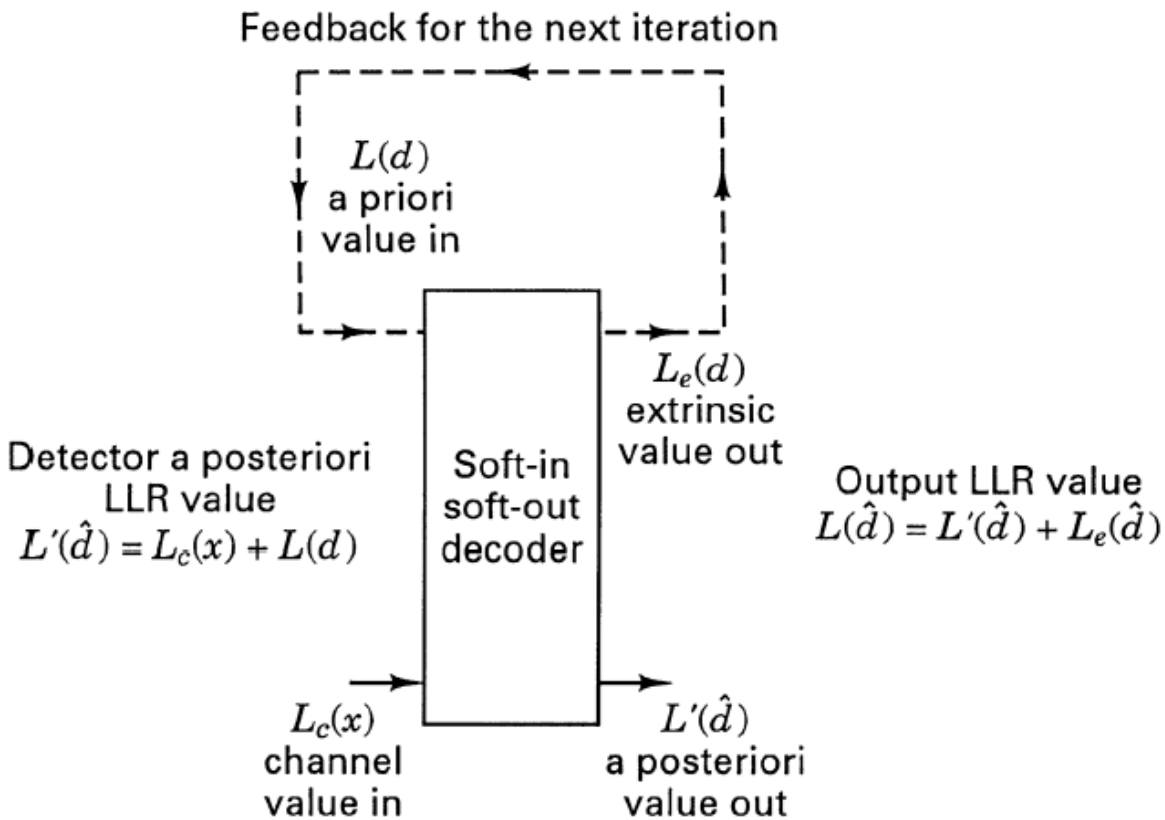
where $L'(\hat{d})$ is the LLR of a data bit out of the demodulator (input to the decoder), and $L_e(\hat{d})$, called the *extrinsic* LLR, represents extra knowledge gleaned from the decoding process. The output sequence of a systematic decoder is made up of values representing data bits and parity bits. From Equations (9) and (10), the output LLR $L(\hat{d})$ of the decoder is now written as follows:

$$L(\hat{d}) = L_c(x) + L(d) + L_e(\hat{d}) \tag{11}$$

Theory of Iterative Coding(3)

Equation (11) shows that the output LLR of a systematic decoder can be represented as having three LLR elements—a channel measurement, a priori knowledge of the data, and an extrinsic LLR stemming solely from the decoder. To yield the final $L(\hat{d})$, each of the individual LLRs can be added as shown in Equation (11), because the three terms are statistically independent [3, 5]. This soft decoder output $L(\hat{d})$ is a real number that provides a hard decision as well as the reliability of that decision. The sign of $L(\hat{d})$ denotes the hard decision; that is, for positive values of $L(\hat{d})$ decide that $d = +1$, and for negative values decide that $d = -1$. The magnitude of $L(\hat{d})$ denotes the reliability of that decision. Often, the value of $L_e(\hat{d})$ due to the decoding has the same sign as $L_c(x) + L(d)$, and therefore acts to improve the reliability of $L(\hat{d})$.

Theory of Iterative Coding (4)



Log Likelihood Algebra

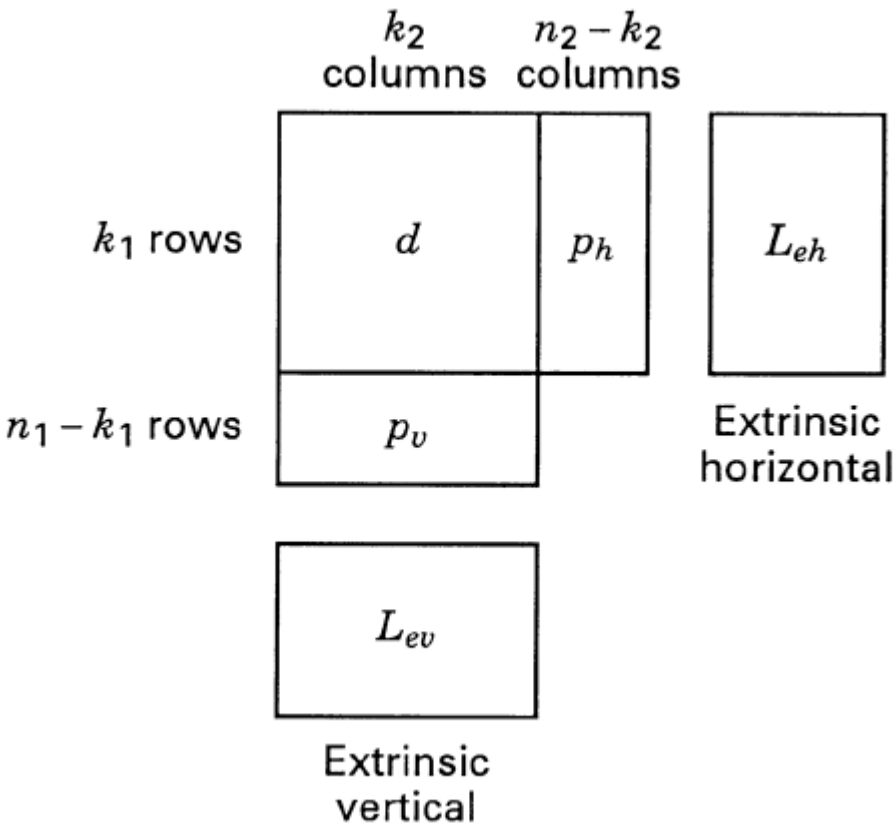
New Operator

Modulo 2 Addition

$$L(d_1) \boxplus L(d_2) \triangleq L(d_1 \oplus d_2) = \log_e \left[\frac{e^{L(d_1)} + e^{L(d_2)}}{1 + e^{L(d_1)} e^{L(d_2)}} \right] \quad (12)$$

$$\approx (-1) \times \text{sgn}\,[L(d_1)] \times \text{sgn}\,[L(d_2)] \times \min\,(|L(d_1)|, |L(d_2)|) \quad (13)$$

Example: Product Code



Iterative Product Decoding

1. Set the a priori LLR $L(d) = 0$ (unless the a priori probabilities of the data bits are other than equally likely).

2. Decode horizontally, and using Equation (11) obtain the horizontal extrinsic LLR as shown below:

$$L_{eh}(\hat{d}) = L(\hat{d}) - L_c(x) - L(d)$$

3. Set $L(d) = L_{eh}(\hat{d})$ for the vertical decoding of step 4.

4. Decode vertically, and using Equation (11) obtain the vertical extrinsic LLR as shown below:

5. Set $L(d) = L_{ev}(\hat{d})$ and repeat steps 2 through 5.

6. After enough iterations (that is, repetitions of steps 2 through 5) to yield a reliable decision, go to step 7.

7. The soft output is

$$L(\hat{d}) = L_c(x) + L_{eh}(\hat{d}) + L_{ev}(\hat{d}) \qquad (14)$$

$$d_i \oplus d_j = p_{ij} \tag{15}$$

$$d_i = d_j \oplus p_{ij} \qquad i, j = 1, 2 \tag{16}$$

where $\oplus$ denotes modulo-2 addition. The transmitted bits are represented by the sequence $d_1\ d_2\ d_3\ d_4\ p_{12}\ p_{34}\ p_{13}\ p_{24}$. At the receiver input, the noise-corrupted bits are represented by the sequence $\{x_i\}$, $\{x_{ij}\}$, where $x_i = d_i + n$ for each received data bit, $x_{ij} = p_{ij} + n$ for each received parity bit, and n represents the noise contribution that is statistically independent for both d_i and p_{ij}. The indices i and j represent

$$L_c(x_k) = \log_e \left[\frac{p(x_k \mid d_k = +1)}{p(x_k \mid d_k = -1)} \right] \tag{17a}$$

$$= \log_e \left(\frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x_k - 1}{\sigma} \right)^2 \right]}{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x_k + 1}{\sigma} \right)^2 \right]} \right) \tag{17b}$$

$$= -\frac{1}{2}\left(\frac{x_k - 1}{\sigma} \right)^2 + \frac{1}{2}\left(\frac{x_k + 1}{\sigma} \right)^2 = \frac{2}{\sigma^2} x_k \tag{17c}$$

$$L_c(x_k) = 2x_k \tag{18}$$

Consider the following example, where the data sequence $d_1\ d_2\ d_3\ d_4$ is made up of the binary digits 1 0 0 1, as shown in Figure 4. By the use of Equation (15), it is seen that the parity sequence $p_{12}\ p_{34}\ p_{13}\ p_{24}$ must be equal to the digits 1 1 1 1. Thus, the transmitted sequence is

$$\{d_i\},\ \{p_{ij}\} = 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \tag{19}$$

When the data bits are expressed as bipolar voltage values of +1 and -1 corresponding to the binary logic levels 1 and 0, the transmitted sequence is

$$\{d_i\},\ \{p_{ij}\} = +1\ -1\ -1\ +1\ +1\ +1\ +1\ +1$$

Assume now that the noise transforms this data-plus-parity sequence into the received sequence

$$\{x_i\},\ \{x_{ij}\} = 0.75,\ 0.05,\ 0.10,\ 0.15,\ 1.25,\ 1.0,\ 3.0,\ 0.5 \tag{20}$$

where the members of $\{x_i\},\ \{x_{ij}\}$ positionally correspond to the data and parity $\{d_i\},\ \{p_{ij}\}$ that was transmitted. Thus, in terms of the positional subscripts, the received sequence can be denoted as

$$\{x_i\},\ \{x_{ij}\} = x_1, x_2, x_3, x_4, x_{12}, x_{34}, x_{13}, x_{24}$$

From Equation (18), the assumed channel measurements yield the LLR values

$$\{L_c(x_i)\},\ \{L_c(x_{ij})\} = 1.5, 0.1, 0.20, 0.3, 2.5, 2.0, 6.0, 1.0 \qquad (21)$$

These values are shown in Figure 4b as the decoder input measurements. It should be noted that, given equal prior probabilities for the transmitted data, if hard decisions are made based on the $\{x_k\}$ or the $\{L_c(x_k)\}$ values shown above, such a process would result in two errors, since d_2 and d_3 would each be incorrectly classified as binary 1.

$d_1 = 1$	$d_2 = 0$	$p_{12} = 1$
$d_3 = 0$	$d_4 = 1$	$p_{34} = 1$
$p_{13} = 1$	$p_{24} = 1$	

(a) Encoder output binary digits

$L_c(x_1) = 1.5$	$L_c(x_2) = 0.1$	$L_c(x_{12}) = 2.5$
$L_c(x_3) = 0.2$	$L_c(x_4) = 0.3$	$L_c(x_{34}) = 2.0$
$L_c(x_{13}) = 6.0$	$L_c(x_{24}) = 1.0$	

(b) Decoder input log-likelihood ratios $L_c(x)$

For the product-code example in Figure 4, we use Equation (11) to express the soft output $L(\hat{d}_1)$ for the received signal corresponding to data d_1, as follows.

$$L(\hat{d}_1) = L_c(x_1) + L(d_1) + \{[L_c(x_2) + L(d_2)] \boxplus L_c(x_{12})\} \quad (22)$$

where the terms $\{[L_c(x_2) + L(d_2)] \boxplus L_c(x_{12})\}$ represent the extrinsic LLR contributed by the code (that is, the reception corresponding to data d_2 and its a priori probability, in conjunction with the reception corresponding to parity p_{12}). In general, the soft output $L(\hat{d}_i)$ for the received signal corresponding to data d_i is

$$L(\hat{d}_i) = L_c(x_i) + L(d_i) + \{[L_c(x_j) + L(d_j)] \boxplus L_c(x_{ij})\} \quad (23)$$

where $L_c(x_i)$, $L_c(x_j)$, and $L_c(x_{ij})$ are the channel LLR measurements of the reception corresponding to d_i, d_j, and p_{ij}, respectively. $L(d_i)$ and $L(d_j)$ are the LLRs of the a priori probabilities of d_i and d_j, respectively, and $\{[L_c(x_j) + L(d_j)] \boxplus L_c(x_{ij})\}$ is the extrinsic LLR contribution from the code. Equations (22) and (23) can best be understood in the context of Figure 4b. For this example, assuming equally-likely signaling, the soft output $L(\hat{d}_1)$ is represented by the detector LLR measurement of $L_c(x_1) = 1.5$ for the reception corresponding to data d_1, plus the extrinsic LLR of $[L_c(x_2) = 0.1] \boxplus [L_c(x_{12}) = 2.5]$ gleaned from the fact that the data d_2 and the parity p_{12} also provide knowledge about the data d_1, as seen from Equations (15) and (16).

For the example in Figure 4, the horizontal calculations for $L_{eh}(\hat{d})$ and the vertical calculations for $L_{ev}(\hat{d})$ are expressed as follows:

$$L_{eh}(\hat{d}_1) = [L_c(x_2) + L(\hat{d}_2)] \boxplus L_c(x_{12}) \tag{24a}$$

$$L_{ev}(\hat{d}_1) = [L_c(x_3) + L(\hat{d}_3)] \boxplus L_c(x_{13}) \tag{24b}$$

$$L_{eh}(\hat{d}_2) = [L_c(x_1) + L(\hat{d}_1)] \boxplus L_c(x_{12}) \tag{25a}$$

$$L_{ev}(\hat{d}_2) = [L_c(x_4) + L(\hat{d}_4)] \boxplus L_c(x_{24}) \tag{25b}$$

$$L_{eh}(\hat{d}_3) = [L_c(x_4) + L(\hat{d}_4)] \boxplus L_c(x_{34}) \tag{26a}$$

$$L_{ev}(\hat{d}_3) = [L_c(x_1) + L(\hat{d}_1)] \boxplus L_c(x_{13}) \tag{26b}$$

$$L_{eh}(\hat{d}_4) = [L_c(x_3) + L(\hat{d}_3)] \boxplus L_c(x_{34}) \tag{27a}$$

$$L_{ev}(\hat{d}_4) = [L_c(x_2) + L(\hat{d}_2)] \boxplus L_c(x_{24}) \tag{27b}$$

The LLR values shown in Figure 4 are entered into the $L_{eh}(\hat{d})$ expressions in Equations (24) through (27) and, assuming equally-likely signaling, the $L(d)$ values are initially set equal to zero, yielding

$$L_{eh}(\hat{d}_1) = (0.1 + 0) \boxplus 2.5 \approx -0.1 = \text{new } L(d_1) \tag{28}$$

$$L_{eh}(\hat{d}_2) = (1.5 + 0) \boxplus 2.5 \approx -1.5 = \text{new } L(d_2) \tag{29}$$

$$L_{eh}(\hat{d}_3) = (0.3 + 0) \boxplus 2.0 \approx -0.3 = \text{new } L(d_3) \tag{30}$$

$$L_{eh}(\hat{d}_4) = (0.2 + 0) \boxplus 2.0 \approx -0.2 = \text{new } L(d_4) \tag{31}$$

where the log-likelihood addition has been calculated using the approximation in Equation (13). Next, we proceed to obtain the first vertical calculations using the $L_{ev}(\hat{d})$ expressions in Equations (24) through (27). Now, the values of $L(d)$ can be refined by using the new $L(d)$ values gleaned from the first horizontal calculations, shown in Equations (28) through (31). That is,

$$L_{ev}(\hat{d}_1) = (0.2 - 0.3) \boxplus 6.0 \approx 0.1 = \text{new } L(d_1) \quad (32)$$

$$L_{ev}(\hat{d}_2) = (0.3 - 0.2) \boxplus 1.0 \approx -0.1 = \text{new } L(d_2) \quad (33)$$

$$L_{ev}(\hat{d}_3) = (1.5 - 0.1) \boxplus 6.0 \approx -1.4 = \text{new } L(d_3) \quad (34)$$

$$L_{ev}(\hat{d}_4) = (0.1 - 1.5) \boxplus 1.0 \approx 1.0 = \text{new } L(d_4) \quad (35)$$

The results of the first full iteration of the two decoding steps (horizontal and vertical) are shown below.

Original $L_c(x_k)$ measurements

1.5	0.1
0.2	0.3

-0.1	-1.5
-0.3	-0.2

$L_{eh}(\hat{d})$ after first horizontal decoding

0.1	-0.1
-1.4	1.0

$L_{ev}(\hat{d})$ after first vertical decoding

Equation (14). The original LLR plus the horizontal extrinsic LLRs yields the following improvement (the extrinsic vertical terms are not yet being considered):

Improved LLRs due to $L_{eh}(\hat{d})$

1.4	-1.4
-0.1	0.1

The original LLR plus both the horizontal and vertical extrinsic LLRs yield the following improvement:

Improved LLRs due to $L_{eh}(\hat{d}) + L_{ev}(\hat{d})$

1.5	-1.5
-1.5	1.1

$$L_{eh}(\hat{d}_1) = (0.1 - 0.1) \boxplus 2.5 \approx 0 = \text{new } L(d_1) \tag{36}$$

$$L_{eh}(\hat{d}_2) = (1.5 + 0.1) \boxplus 2.5 \approx -1.6 = \text{new } L(d_2) \tag{37}$$

$$L_{eh}(\hat{d}_3) = (0.3 + 1.0) \boxplus 2.0 \approx -1.3 = \text{new } L(d_3) \tag{38}$$

$$L_{eh}(\hat{d}_4) = (0.2 - 1.4) \boxplus 2.0 \approx 1.2 = \text{new } L(d_4) \tag{39}$$

$$L_{ev}(\hat{d}_1) = (0.2 - 1.3) \boxplus 6.0 \approx 1.1 = \text{new } L(d_1) \tag{40}$$

$$L_{ev}(\hat{d}_2) = (0.3 + 1.2) \boxplus 1.0 \approx -1.0 = \text{new } L(d_2) \tag{41}$$

$$L_{ev}(\hat{d}_3) = (1.5 + 0) \boxplus 6.0 \approx -1.5 = \text{new } L(d_3) \tag{42}$$

$$L_{ev}(\hat{d}_4) = (0.1 - 1.6) \boxplus 1.0 \approx 1.0 = \text{new } L(d_4) \tag{43}$$

$$L(\hat{d}) = L_c(x) + L_{eh}(\hat{d}) + L_{ev}(\hat{d}) \quad (44)$$

The horizontal and vertical extrinsic LLRs of Equations (36) through (43) and the resulting decoder LLRs are displayed below. For this example, the second horizontal and vertical iteration (yielding a total of four iterations) suggests a modest improvement over a single horizontal and vertical iteration. The results show a balancing of the confidence values among each of the four data decisions.

Original $L_c(x)$ measurements

1.5	0.1
0.2	0.3

0	-1.6
-1.3	1.2

$L_{eh}(\hat{d})$ after second horizontal decoding

1.1	-1.0
-1.5	1.0

$L_{ev}(\hat{d})$ after second vertical decoding

The soft output is

$$L(\hat{d}) = L_c(x) + L_{eh}(\hat{d}) + L_{ev}(\hat{d})$$

which after a total of four iterations yields the values for $L(\hat{d})$ of

2.6	-2.5
-2.6	2.5

Observe that correct decisions about the four data bits will result, and the level of confidence about these decisions is high. The iterative decoding of turbo codes is similar to the process used when solving a crossword puzzle. The first pass through the puzzle is likely to contain a few errors. Some words seem to fit, but when the letters intersecting a row and column do not match, it is necessary to go back and correct the first-pass answers.

RSC vs NSC

. It is well known that at large E_b/N_0 values, the error performance of an NSC is better than that of a systematic code having the same memory. At small E_b/N_0 values, it is generally the other way around [3]. A class of infinite impulse response (IIR) convolutional codes [3] has been proposed as building blocks for a turbo code. Such building blocks are also referred to as

recursive systematic convolutional (RSC) codes because previously encoded information bits are continually fed back to the encoder's input. For high code rates, RSC codes result in better error performance than the best NSC codes at any value of E_b/N_0. A binary, rate 1/2, RSC code is obtained from an NSC code by using a feedback loop and setting one of the two outputs (u_k or v_k) equal to d_k. Figure 6(a) illustrates an example of such an RSC code, with $K = 3$, where a_k is recursively calculated as

$$a_k = d_k + \sum_{i=1}^{K-1} g'_i \, a_{k-i} \mod 2 \qquad (47)$$

and g'_i is respectively equal to g_{1i} if $u_k = d_k$, and to g_{2i} if $v_k = d_k$. Figure 6(b) shows the trellis structure for the RSC code in Figure 6(a).

Recursive/Systematic Convolutional Coding

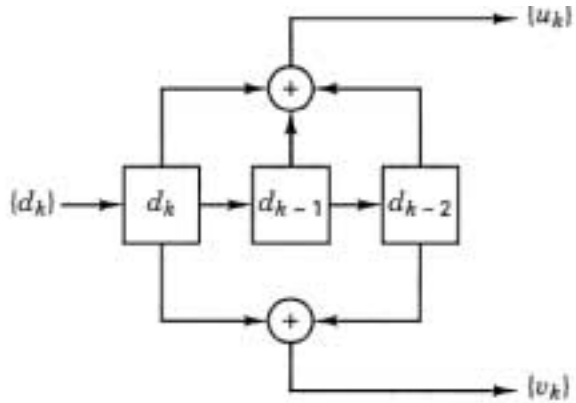


Figure 5

Nonsystematic convolutional (NSC) code.

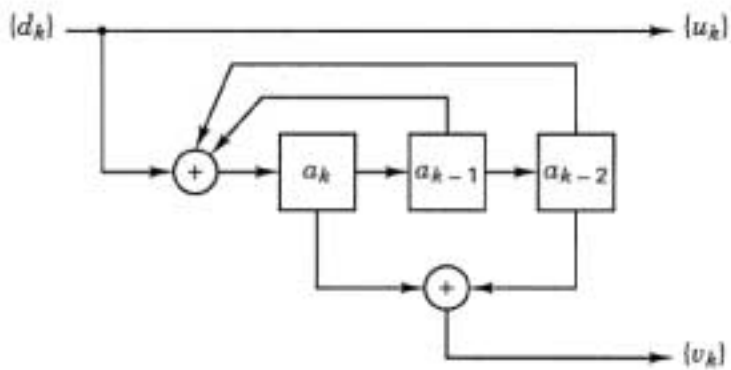


Figure 6(a)

Recursive systematic convolutional (RSC) code.

Table 1

Validation of the Figure 6(b) Trellis Section

Input bit $d_k = u_k$	Current bit a_k	Starting state		Code bits $u_k v_k$	Ending state	
		a_{k-1}	a_{k-2}		a_k	a_{k-1}
0	0	0	0	0 0	0	0
	1	1	0	0 1	1	1
	1	0	1	0 0	1	0
	0	1	1	0 1	0	1
1	1	0	0	1 1	1	0
	0	1	0	1 0	0	1
	0	0	1	1 1	0	0
	1	1	1	1 0	1	1

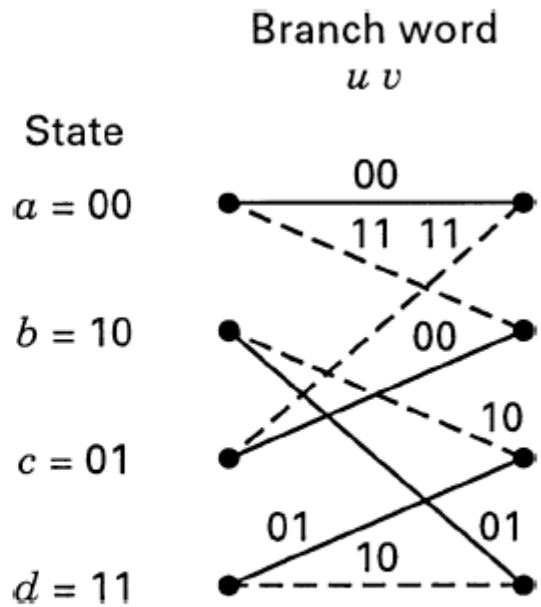
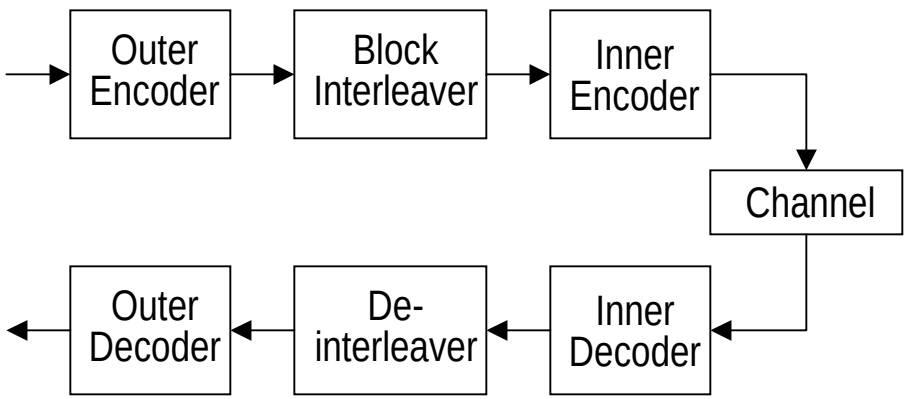


Figure 6(b)

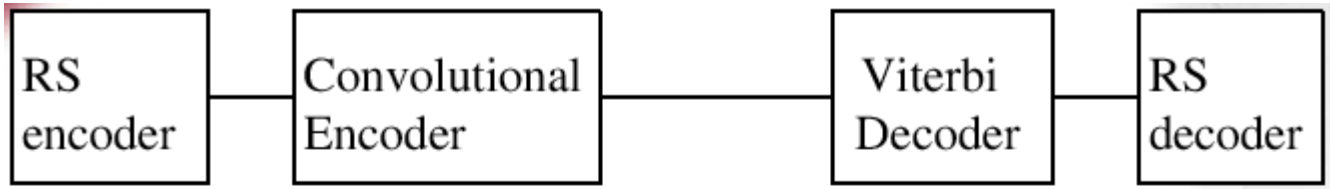
Trellis structure for the RSC code in Figure 6(a).

Concatenated Coding

- A single error correction code does not always provide enough error protection with reasonable complexity.
- Solution: Concatenate two (or more) codes
  - This creates a much more powerful code.
- Serial Concatenation (Forney, 1966)



Concatenated Codes (2)



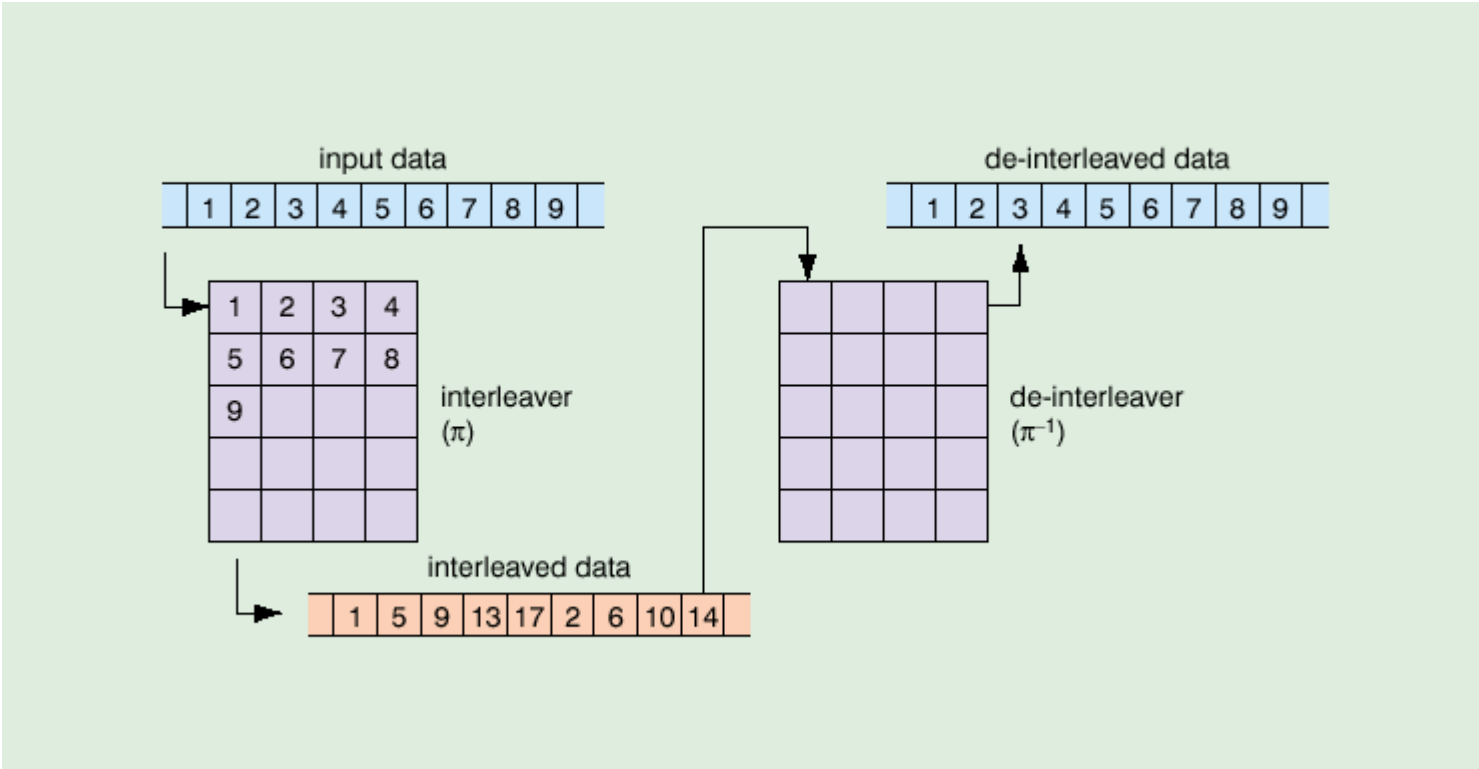
- Traditional scheme for very low error rate
- Typically RS + Convolutional
- Viterbi - soft decisions
- RS burst error correction
- Viterbi gives moderate BER where RS takes over and brings it further down
- Long Code with practical complexity

Alternative to Concatenated Coding

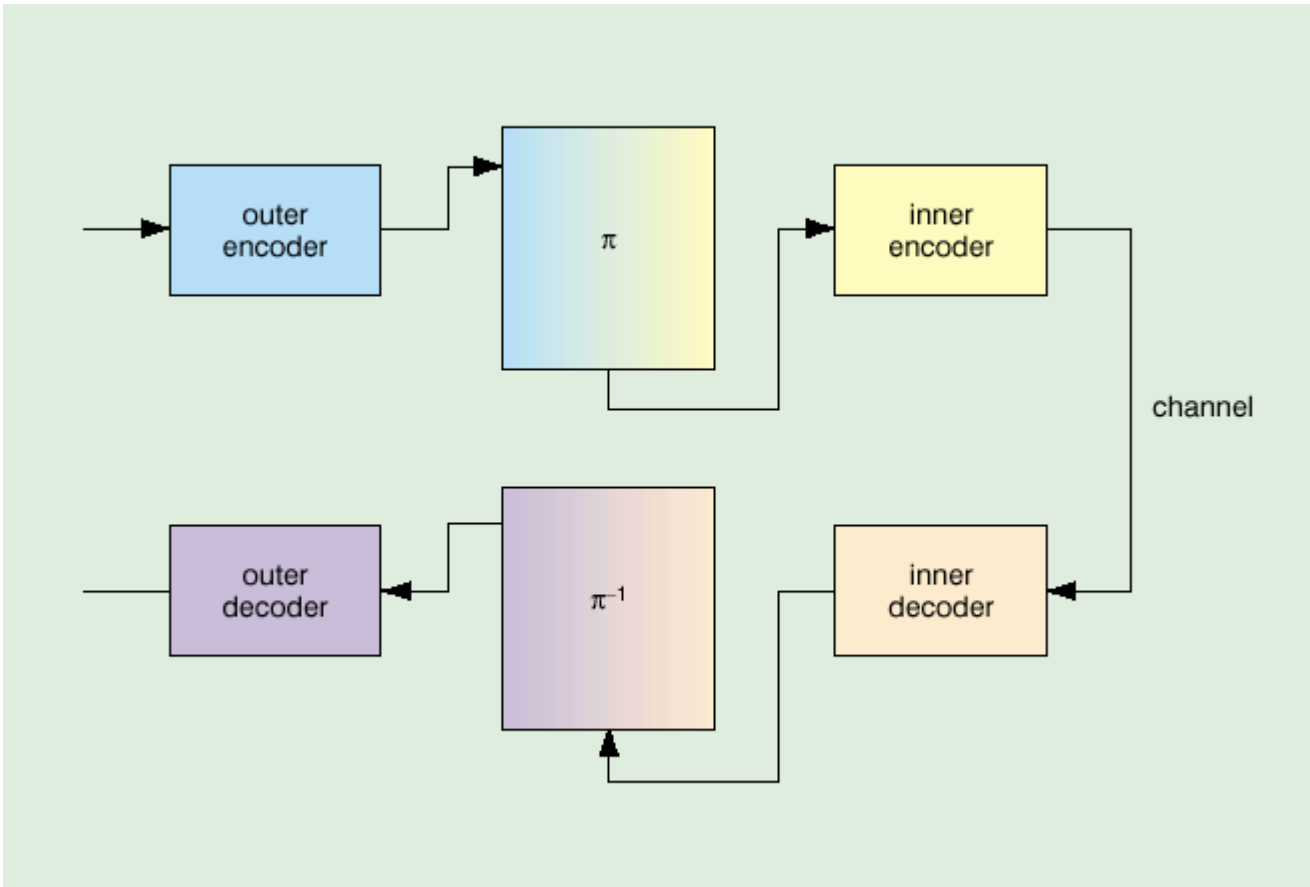
- Use a very long constraint length (K) convolutional code
- Typically - K=30-50
- Error rate decreases exponentially with constraint length
- Viterbi decoding:
  - $2^{30} \sim 10^9$ states
- NOT IMPLEMENTABLE

- Solution: Sequential Decoding
  - Limited but smart search
  - Search the most likely paths
  - Complexity depends on channel conditions
  - Can calculate conditions for when decoding is possible

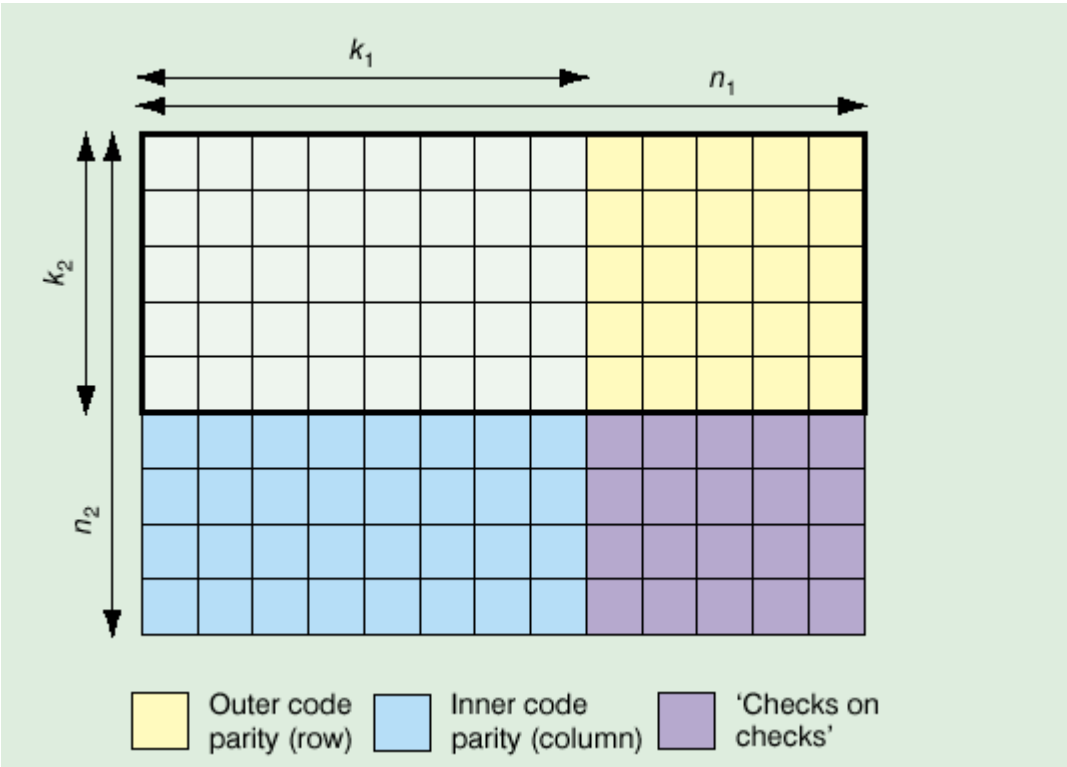
Interleaver/Deinterleaver



Concatenated interleaver and De-interleaver



Structure of Concatenated Interleaver system



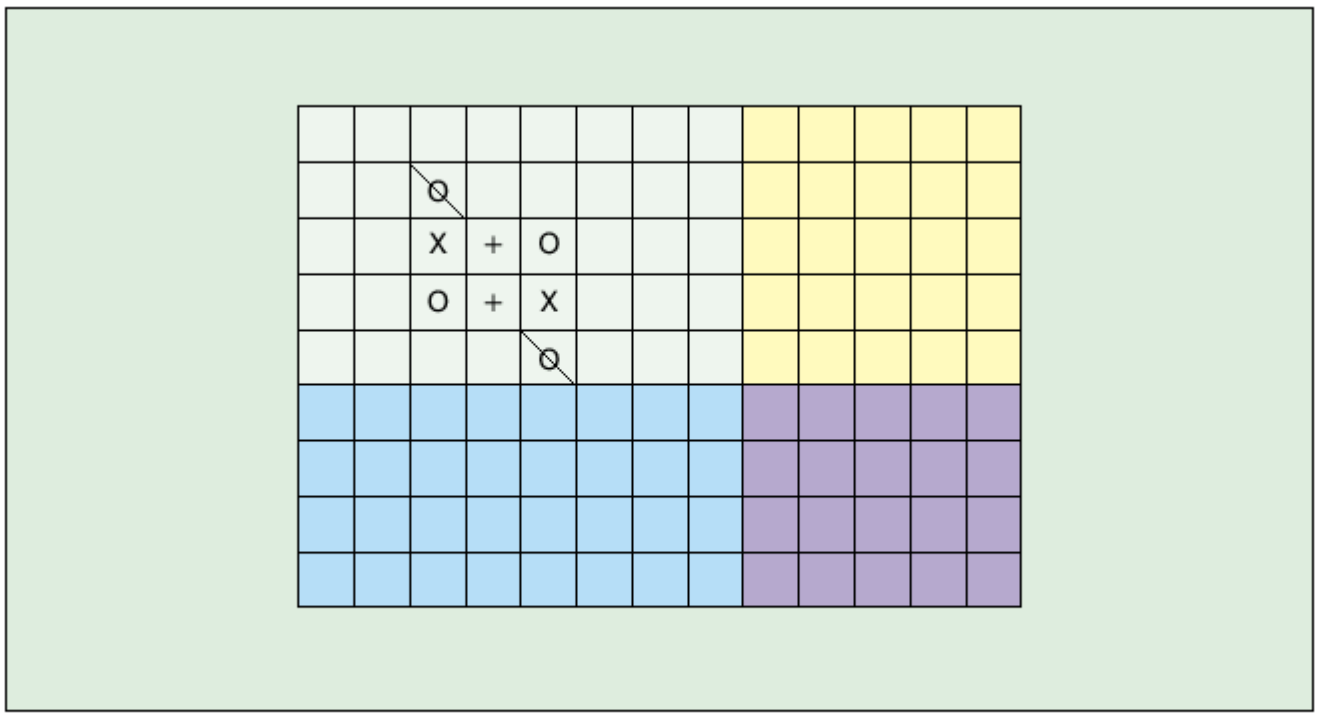
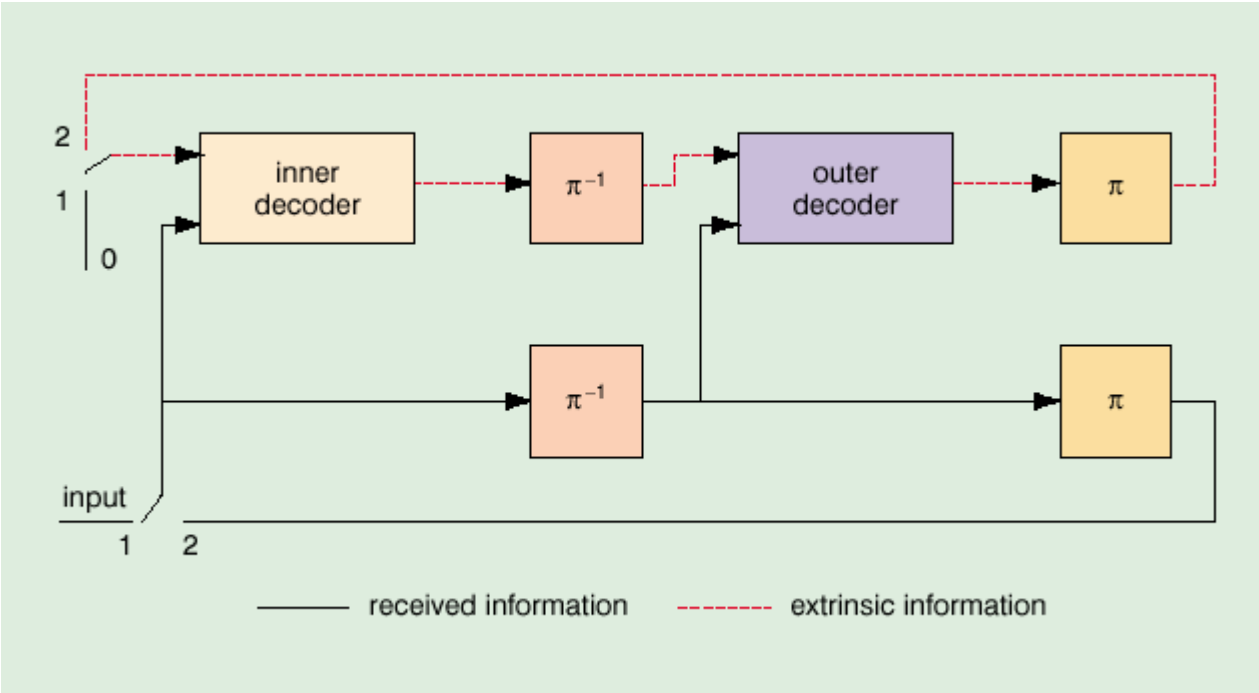


Fig. 6 Pattern of received errors ('O') in codeword array, with errors introduced by inner (column) decoder ('X') and outer (row) decoder ('+')

Iterative concatenated decoding



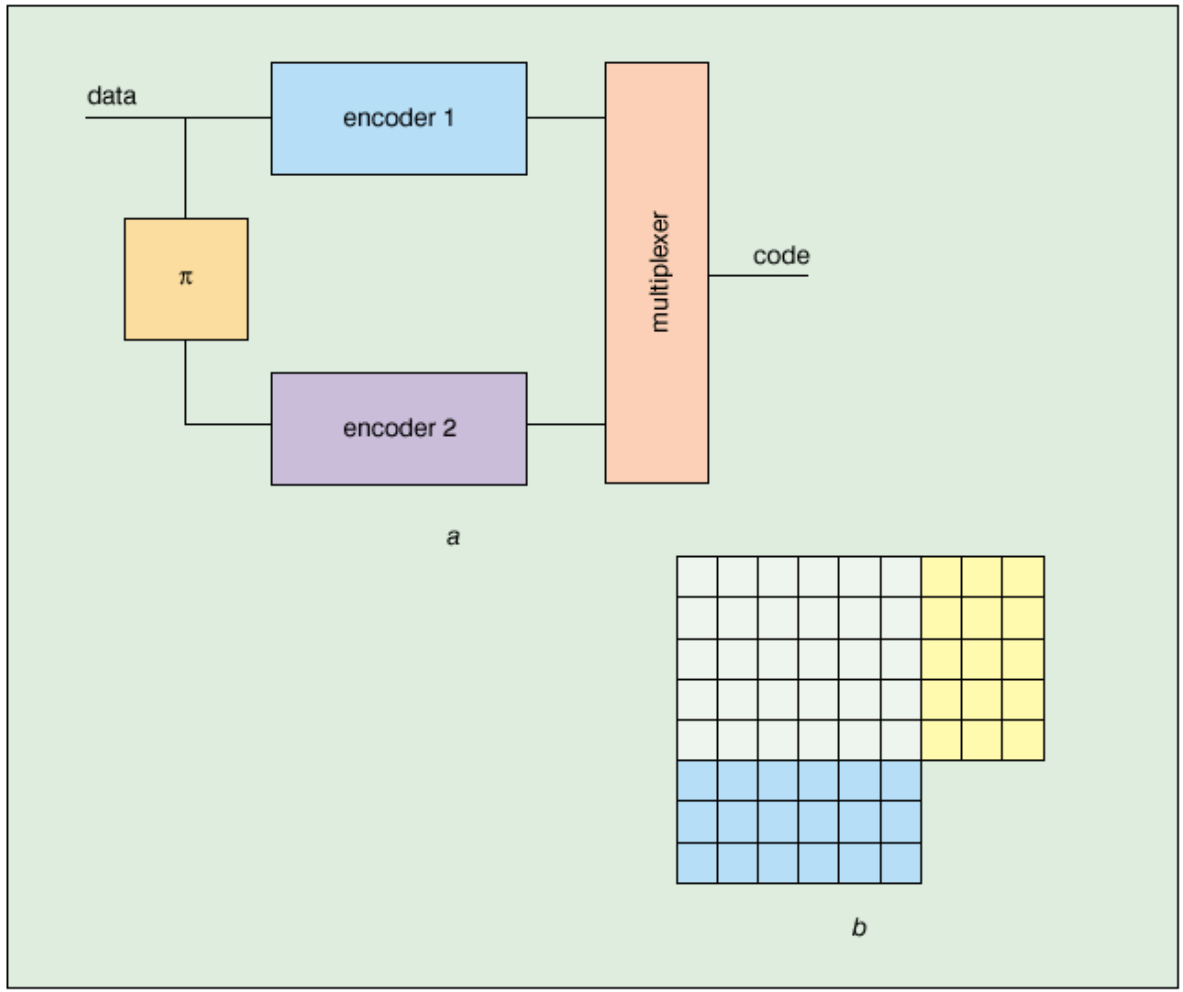


Fig. 8 Parallel concatenation: (*a*) encoder structure; (*b*) code array

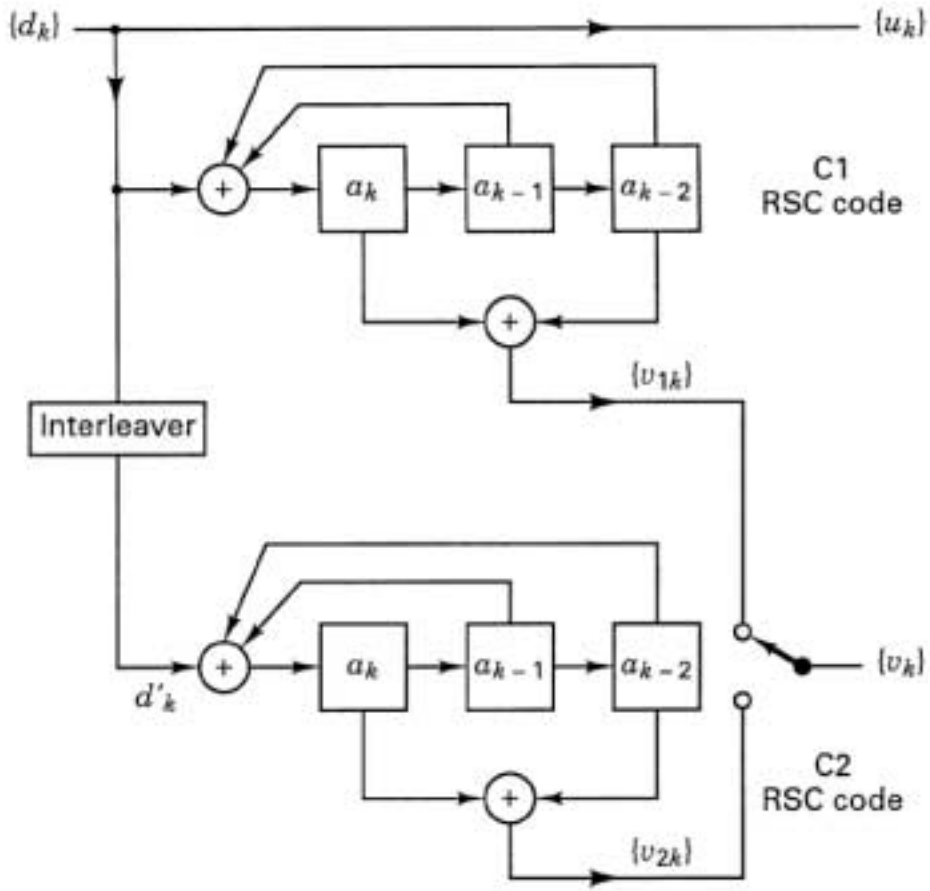


Figure 7

Parallel concatenation of two RSC codes.

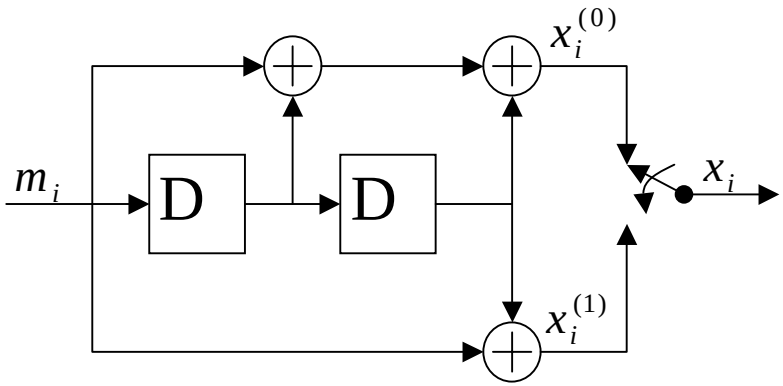
Turbo Codes

- Backgound
  - Turbo codes were proposed by Berrou and Glavieux in the 1993 International Conference in Communications.
  - Performance within 0.5 dB of the channel capacity limit for BPSK was demonstrated.
- Features of turbo codes
  - Parallel concatenated coding
  - Recursive convolutional encoders
  - Pseudo-random interleaving
  - Iterative decoding

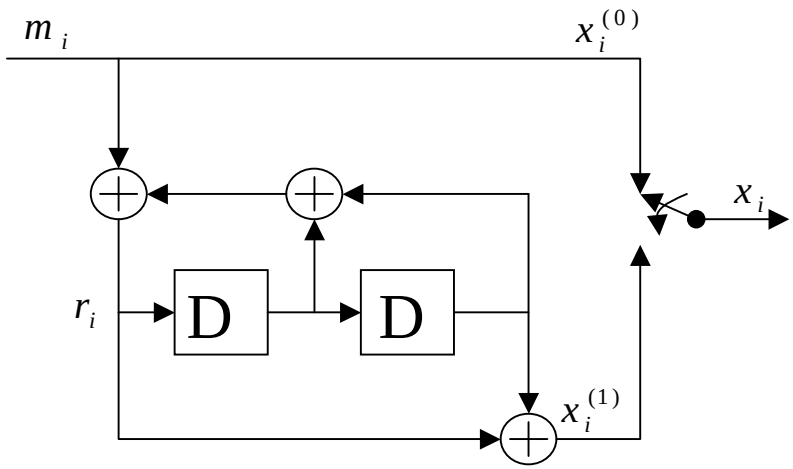
The building blocks of turbo codes

- Recursive systematic codes
- Parallel Concatenation plus puncturing
- Interleaving

Recursive Systematic Convolutional Encoding



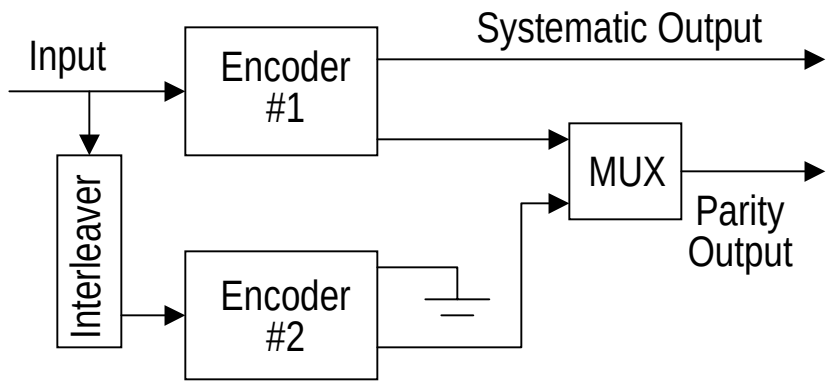
Constraint Length K= 3



- An RSC encoder can be constructed from a standard convolutional encoder by feeding back one of the outputs.
- An RSC encoder has an infinite impulse response.
- An arbitrary input will cause a “good” (high weight) output with high probability.
- Some inputs will cause “bad” (low weight) outputs.

Parallel Concatenated Codes

- Instead of concatenating in serial, codes can also be concatenated in parallel.
- The original turbo code is a parallel concatenation of two *recursive systematic convolutional* (RSC) codes.
  - ***systematic***: one of the outputs is the input.



Parallel Concatenation of RSC codes

Consider the parallel concatenation of two RSC encoders of the type shown in Figure 6. Good turbo codes have been constructed from component codes having short constraint lengths (K = 3 to 5). An example of a such a turbo encoder is shown in Figure 7, where the switch yielding v_k provides puncturing, making the overall code rate 1/2. Without the switch, the code would be rate 1/3. There is no limit to the number of encoders that may be concatenated, and in general the component codes need not be identical with regard to constraint length and rate. The goal in designing turbo codes is to choose the best component codes by maximizing the effective free distance of the code [7]. At large values of E_b/N_0, this is tantamount to maximizing the minimum-weight codeword. However, at low values of E_b/N_0 (the region of greatest interest), optimizing the weight distribution of the codewords is more important than maximizing the minimum-weight codeword [6].

The turbo encoder in Figure 7 produces codewords from each of two component encoders. The weight distribution for the codewords out of this parallel concatenation depends on how the codewords from one of the component encoders are combined with codewords from the other encoder. Intuitively, we should avoid pairing low-weight codewords from one encoder with low-weight codewords from the other encoder. Many such pairings can be avoided by proper design of the interleaver. An interleaver that permutes the data in a random fashion provides better performance than the familiar block interleaver [8].

If the component encoders are not recursive, the unit weight input sequence 0 0 … 0 0 1 0 0 … 0 0 will always generate a low-weight codeword at the input of a second encoder for any interleaver design. In other words, the interleaver would not influence the output-codeword weight distribution if the component codes were not recursive. However if the component codes are recursive, a weight-1 input sequence generates an infinite impulse response (infinite-weight output). Therefore, for the case of recursive codes, the weight-1 input sequence does not yield the minimum-weight codeword out of the encoder. The encoded output weight is kept finite only by trellis termination, a process that forces the coded

sequence to terminate in such a way that the encoder returns to the zero state. In effect, the convolutional code is converted to a block code.

For the encoder of Figure 7, the minimum-weight codeword for each component encoder is generated by the weight-3 input sequence (0 0 … 0 0 1 1 1 0 0 0 … 0 0) with three consecutive 1s. Another input that produces fairly low-weight codewords is the weight-2 sequence (0 0 … 0 0 1 0 0 1 0 0 … 0 0). However, after the permutations introduced by an interleaver, either of these deleterious input patterns is unlikely to appear again at the input to another encoder, making it unlikely that a minimum-weight codeword will be combined with another minimum-weight codeword.

The important aspect of the building blocks used in turbo codes is that they are recursive (the systematic aspect is merely incidental). It is the RSC code's IIR property that protects against the generation of low-weight codewords that cannot be remedied by an interleaver. One can argue that turbo code performance is largely influenced by minimum-weight codewords that result from the weight-2 input sequence. The argument is that weight-1 inputs can be ignored, since they yield large codeword weights due to the IIR encoder structure. For input sequences having weight-3 and larger, a properly designed interleaver makes the occurrence of low-weight output codewords relatively rare [7-11].

Pseudo-random Interleaving

- The coding dilemma:
  - Shannon showed that large block-length random codes achieve channel capacity.
  - However, codes must have structure that permits decoding with reasonable complexity.
  - Codes with structure don't perform as well as random codes.
  - "Almost all codes are good, except those that we can think of."
- Solution:
  - Make the code appear random, while maintaining enough structure to permit decoding.
  - This is the purpose of the pseudo-random interleaver.
  - Turbo codes possess random-like properties.
  - However, since the interleaving pattern is known, decoding is possible.

Why Interleaving and Recursive Encoding?

- In a coded systems:
  - Performance is dominated by low weight code words.
- A “good” code:
  - will produce low weight outputs with very low probability.
- An RSC code:
  - Produces low weight outputs with fairly low probability.
  - However, some inputs still cause low weight outputs.
- Because of the interleaver:
  - The probability that both encoders have inputs that cause low weight outputs is very low.
  - Therefore the parallel concatenation of both encoders will produce a “good” code.

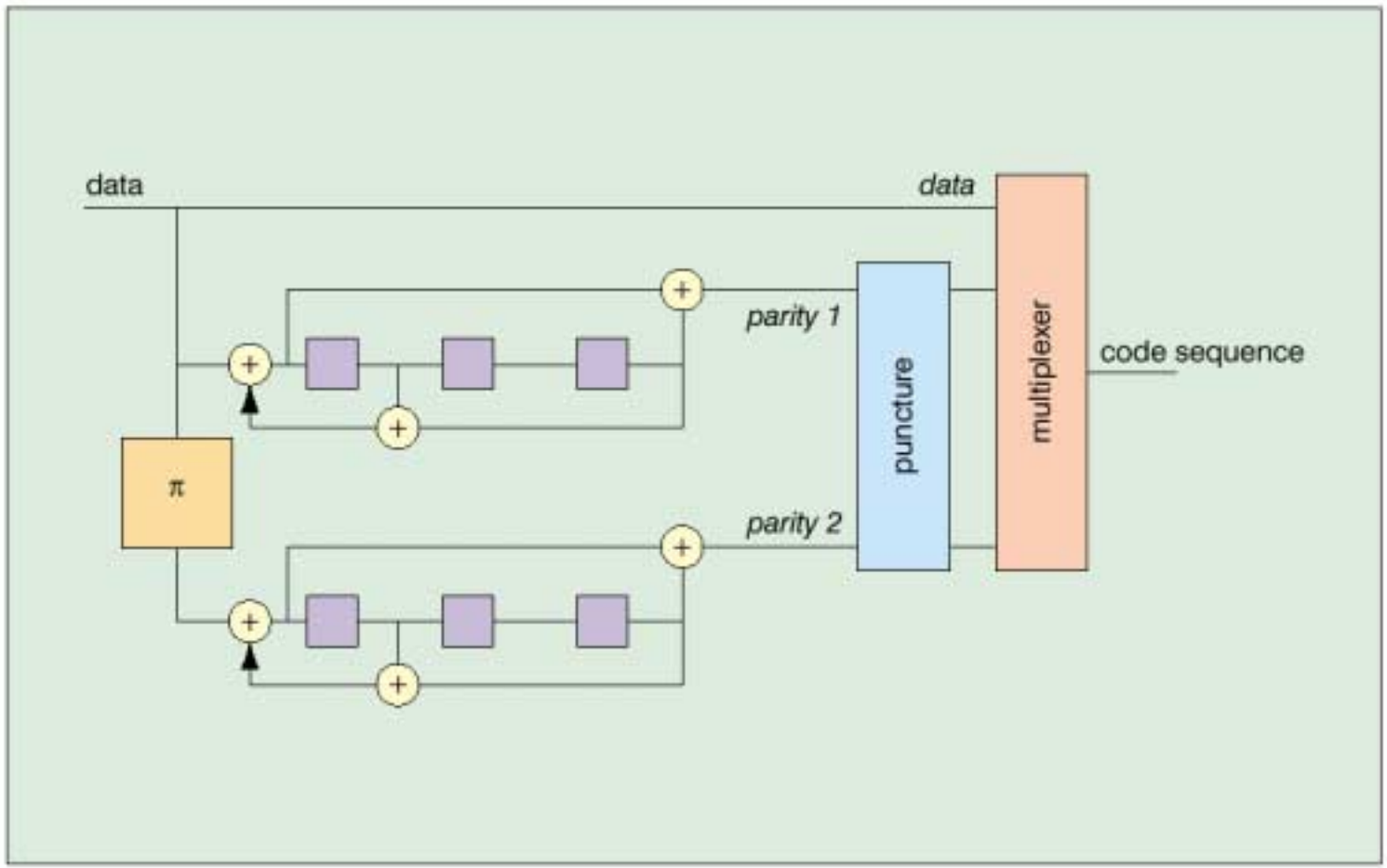


Fig. 11 Turbo-encoder with puncturing

Theory of Turbo-decoding

The Viterbi algorithm (VA) is an optimal decoding method for minimizing the probability of sequence error. Unfortunately, the VA is not suited to generate the a posteriori probability (APP) or soft-decision output for each decoded bit. A relevant algorithm for doing this has been proposed by Bahl et al. [12] The Bahl algorithm was modified by Berrou, et al. [3] for use in decoding RSC codes. The APP of a decoded data bit d_k can be derived from the joint probability $\lambda_k^{i,m}$ defined by

$$\lambda_k^{i,m} = P\{d_k = i, S_k = m | R_1^N\} \tag{48}$$

where S_k is the encoder state at time k, and R_1^N is a received binary sequence from time $k = 1$ through some time N.

Thus, the APP for a decoded data bit d_k, represented as a binary digit, is equal to

$$P\{d_k = i \,|\, R_1^N\} = \sum_m \lambda_k^{i,m} \quad i = 0, 1 \tag{49}$$

Turbo decoding (2)

The log-likelihood ratio (LLR) is written as the logarithm of the ratio of APPs, as follows:

$$L(\hat{d}_k) = \log\left[\frac{\sum_m \lambda_k^{1,m}}{\sum_m \lambda_k^{0,m}}\right] \tag{50}$$

The decoder makes a decision, known as the *maximum a posteriori* (*MAP*) decision rule, by comparing $L(\hat{d}_k)$ to a zero threshold. That is,

$$\begin{aligned} \hat{d}_k &= 1 \quad \text{if} \quad L(\hat{d}_k) > 0 \\ \hat{d}_k &= 0 \quad \text{if} \quad L(\hat{d}_k) < 0 \end{aligned} \tag{51}$$

For a systematic code, the LLR $L(\hat{d}_k)$ associated with each decoded bit $\hat{d}_k$ can be described as the sum of the LLR of $\hat{d}_k$ out of the demodulator and of other LLRs generated by the decoder (extrinsic information), as expressed in Equations (12) and (13). Consider the detection of a noisy data sequence that stems from the encoder of Figure 7, with the use of a decoder shown in Figure 8. Assume binary modulation and a discrete memoryless Gaussian channel. The decoder input is made up of a set R_k of two random variables x_k and y_k. For the bits d_k and v_k at time k, expressed as binary numbers (1, 0), the conversion to received bipolar (+1, -1) pulses can be expressed as follows:

$$x_k = (2d_k - 1) + i_k \tag{52}$$

$$y_k = (2v_k - 1) + q_k \tag{53}$$

where i_k and q_k are two statistically-independent random variables with the same variance σ^2, accounting for the noise contribution. The redundant information, y_k, is demultiplexed and sent to decoder DEC1 as y_{1k} when $v_k = v_{1k}$, and to decoder DEC2 as y_{2k} when $v_k = v_{2k}$. When the redundant information of a given encoder (C1 or C2) is not emitted, the corresponding decoder input is set to zero. Note that the output of DEC1 has an interleaver structure identical to the one used at the transmitter between the two encoders. This is because the information processed by DEC1 is the noninterleaved output of C1 (corrupted by channel noise). Conversely, the information processed by DEC2 is the noisy output of C2 whose input is the same data going into C1, however permuted by the interleaver. DEC2 makes use of the DEC1 output, provided that this output is time-ordered in the same way as the

input to C2 (that is, the two sequences into DEC2 must appear "in step" with respect to the positional arrangement of the signals in each sequence).

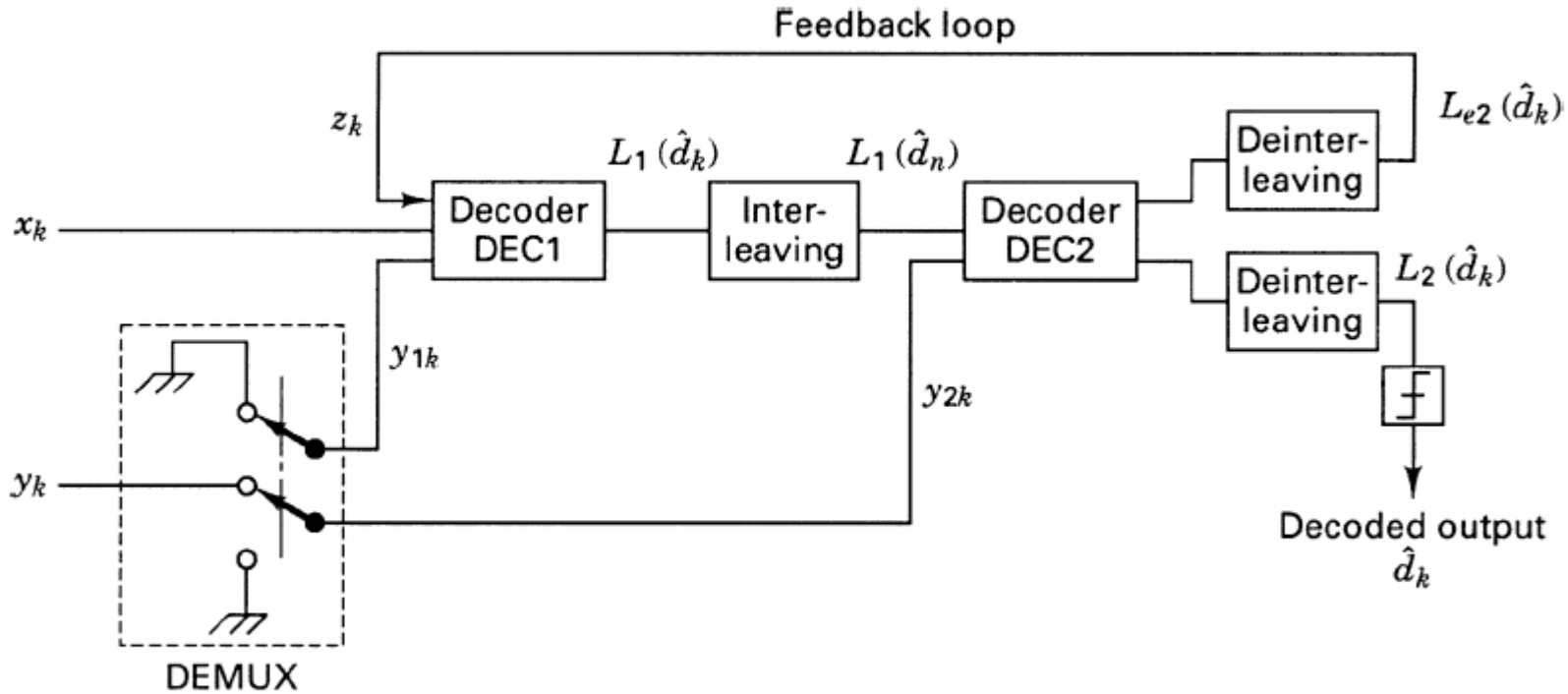
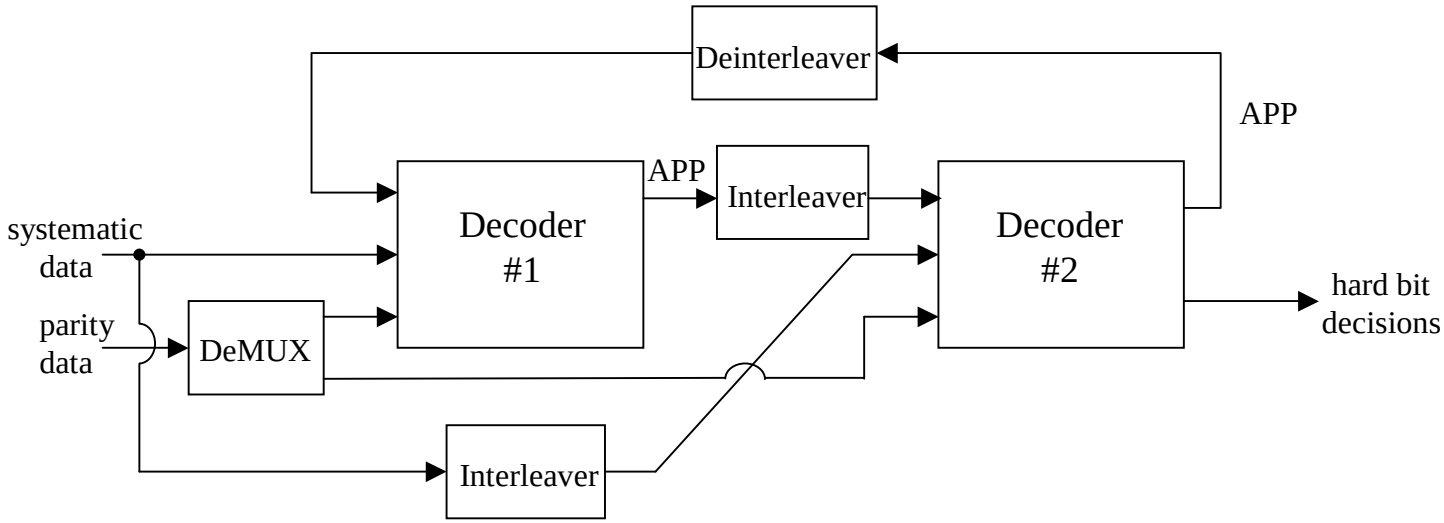


Figure 8

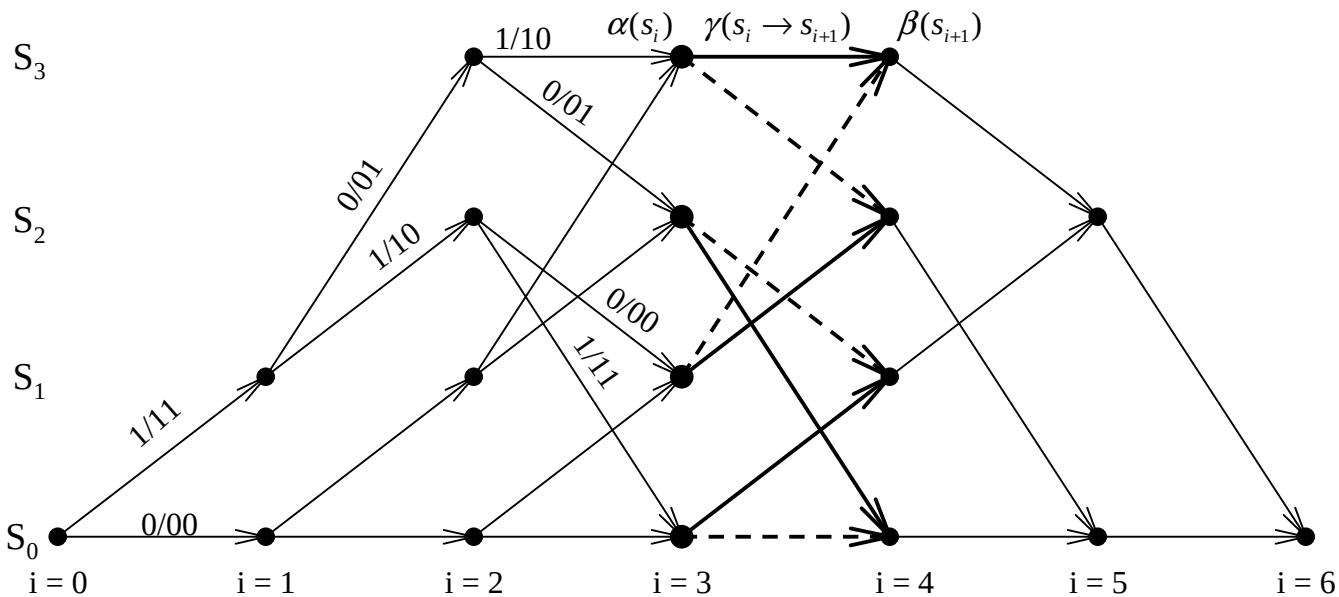
Feedback decoder.

Iterative Decoding



- There is one decoder for each elementary encoder.
- Each decoder estimates the *a posteriori probability* (APP) of each data bit.
- The APP's are used as *a priori* information by the other decoder.
- Decoding continues for a set number of iterations.
  - Performance generally improves from iteration to iteration, but follows a law of diminishing returns.

The log-MAP algorithm



The log-MAP algorithm:

Performs arithmetic in the log domain

Multiplies become additions

Additions use the Jacobian Logarithm:

$$\ln(e^x + e^y) = \max(x, y) + \ln(1 + e^{-|y-x|})$$

Decoding with a feedback loop

We rewrite Equation (11) for the soft-decision output at time k, with the a priori LLR $L(d_k)$ initially set to zero. This follows from the assumption that the data bits are equally likely. Therefore,

$$L(\hat{d}_k) = L_c(x_k) + L_e(\hat{d}_k)$$

$$= \log\left[\frac{p(x_k \mid d_k = 1)}{p(x_k \mid d_k = 0)}\right] + L_e(\hat{d}_k) \qquad (54)$$

where $L(\hat{d}_k)$ is the soft-decision output at the decoder, and $L_c(x_k)$ is the LLR channel measurement, stemming from the ratio of likelihood functions $p(x_k|d_k = i)$ associated with the discrete memoryless channel model. $L_e(\hat{d}_k) = L(\hat{d}_k)|_{x_k = 0}$ is a function of the redundant information. It is the extrinsic information supplied by the decoder, and does not depend on the decoder input x_k. Ideally, $L_c(x_k)$ and $L_e(\hat{d}_k)$ are corrupted by uncorrelated noise, and thus $L_e(\hat{d}_k)$ may be used as a new observation of d_k by another decoder for an iterative process. The fundamental principle for feeding back information to another decoder is that a decoder should never be supplied with information that stems from itself (because the input and output corruption will be highly correlated).

For the Gaussian channel, the natural logarithm in Equation (54) is used to describe the channel LLR, $L_c(x_k)$, as in Equations (17a-c). We rewrite the Equation (17c) LLR result below:

$$L_c(x_k) = -\frac{1}{2}\left(\frac{x_k - 1}{\sigma}\right)^2 + \frac{1}{2}\left(\frac{x_k + 1}{\sigma}\right)^2 = \frac{2}{\sigma^2}\, x_k \tag{55}$$

Both decoders, DEC1 and DEC2, use the modified Bahl algorithm [12]. If the inputs $L_1(\hat{d}_k)$ and y_{2k} to decoder DEC2 are statistically independent, the LLR $L_2(\hat{d}_k)$ at the output of DEC2 can be written as

$$L_2(\hat{d}_k) = f\,[L_1(\hat{d}_k)] + L_{e2}(\hat{d}_k) \tag{56}$$

with

$$L_1(\hat{d}_k) = \frac{2}{\sigma_0^2}\, x_k + L_{e1}(\hat{d}_k) \tag{57}$$

where $f[\bullet]$ indicates a functional relationship. The extrinsic information $L_{e2}(\hat{d}_k)$ out of DEC2 is a function of the sequence $\{L_1(\hat{d}_n)\}_{n\neq k}$. Since $L_1(\hat{d}_n)$ depends on the observation R_1^N, the extrinsic information $L_{e2}(\hat{d}_k)$ is correlated with observations x_k and y_{1k}. Nevertheless, the greater $|n-k|$ is, the less correlated are $L_1(\hat{d}_n)$ and the observations x_k, y_k. Thus, due to the interleaving between DEC1 and DEC2, the extrinsic information $L_{e2}(\hat{d}_k)$ and the observations x_k, y_{1k} are weakly correlated. Therefore, they can be jointly used for the decoding of bit d_k. In Figure 8, the parameter $z_k = L_{e2}(\hat{d}_k)$ feeding into DEC1 acts as a diversity effect in an iterative process. In general, $L_{e2}(\hat{d}_k)$ will have the same sign as d_k. Therefore, $L_{e2}(\hat{d}_k)$ may increase the associated LLR and thus improve the reliability of each decoded data bit.

The algorithmic details for computing the LLR, $L(\hat{d}_k)$, of the a posteriori probability (APP) for each data bit has been described by several authors [3-4,16]. Suggestions for decreasing the implementational complexity of the algorithms can be found in [13-17]. A reasonable way to think of the process that produces APP values for each data bit is to imagine implementing a maximum-likelihood

Iterative Turbo Decoder

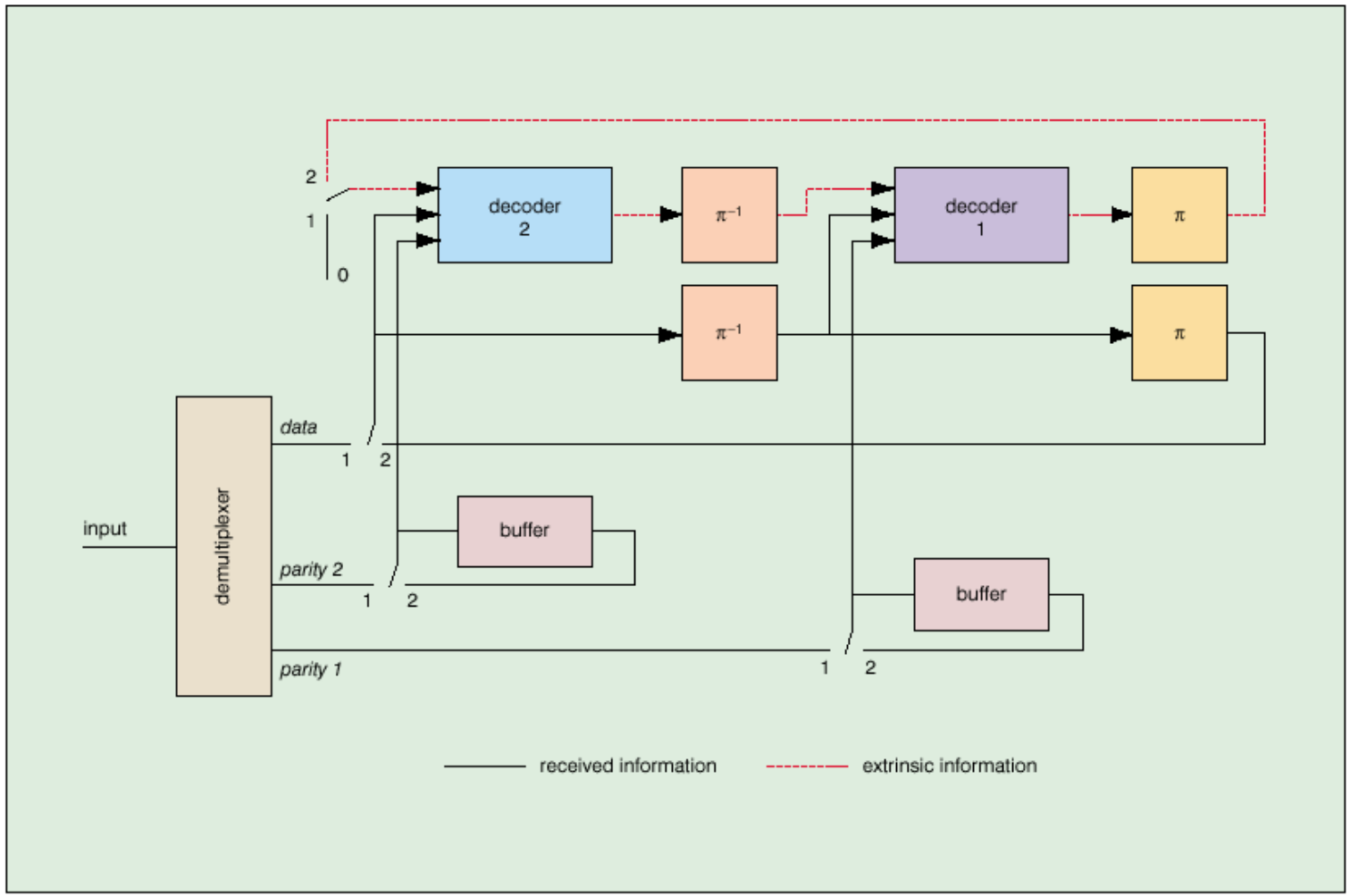
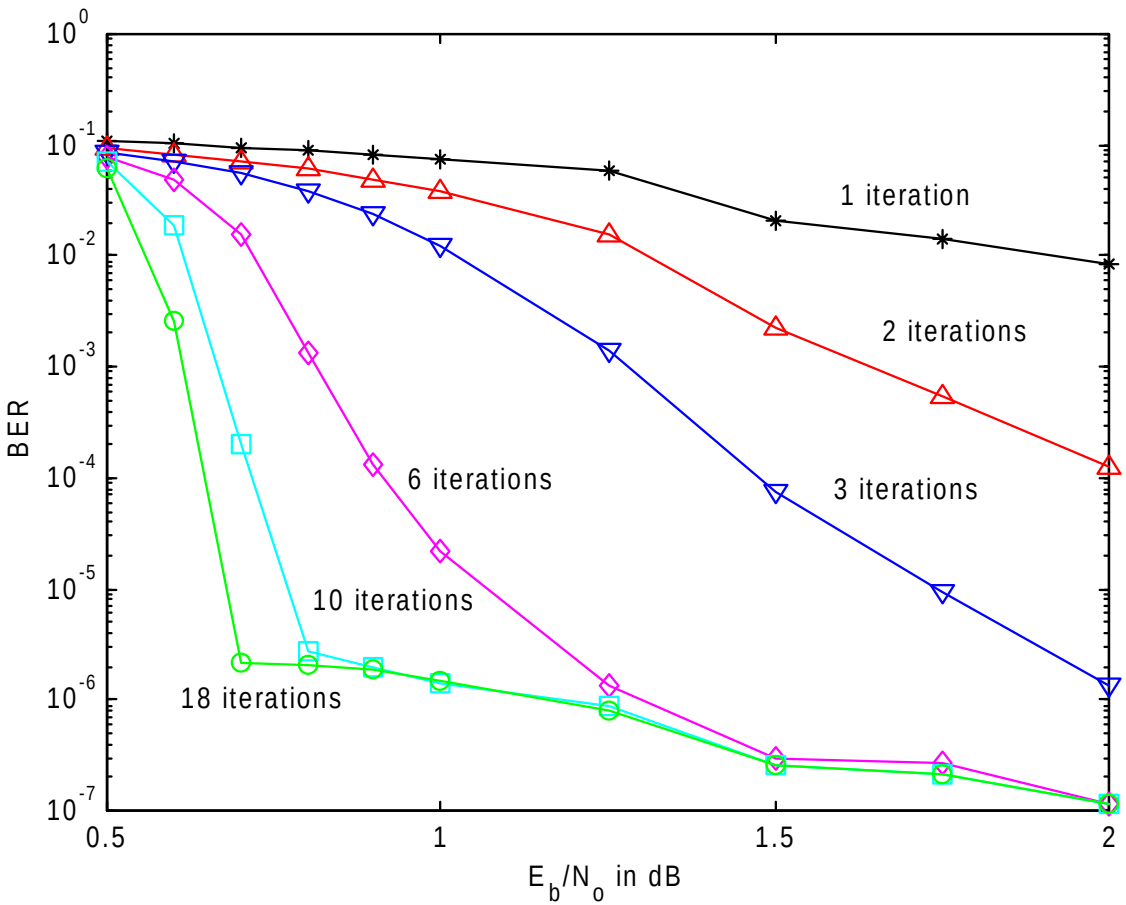


Fig. 12 Iterative turbo-decoder

Performance as a Function of Number of Iterations



- K=5
- r=1/2
- L=65,536

Another Example

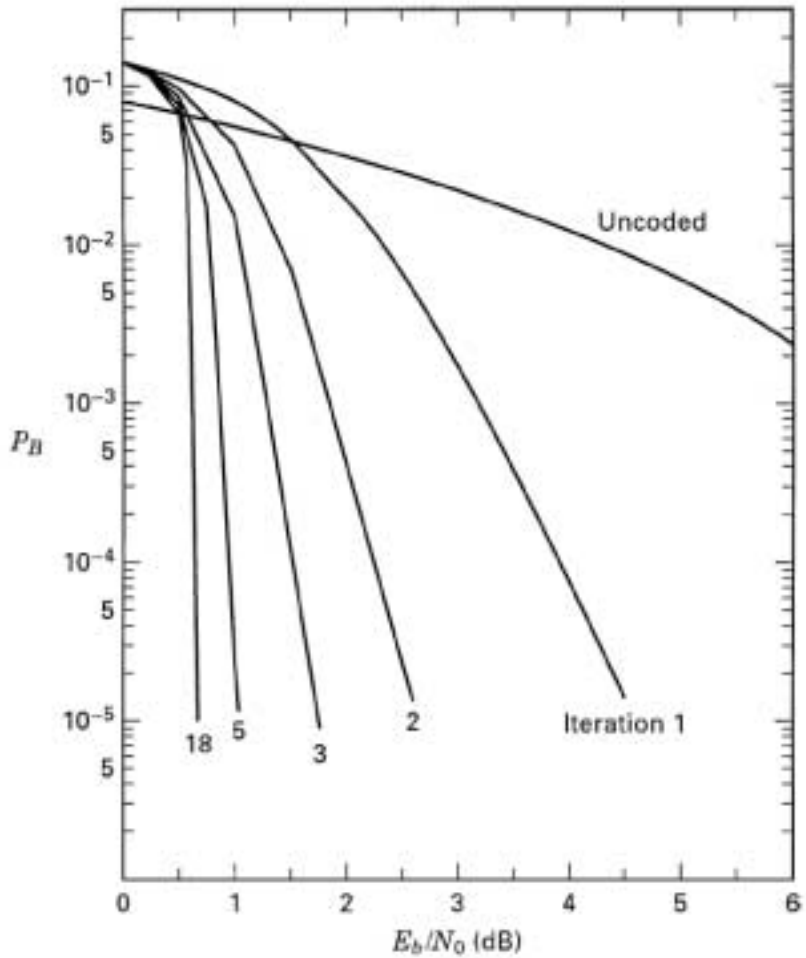


Figure 9

Bit-error probability as a function of E_b/N_0 and multiple iterations.

Performance Factors and Tradeoffs

- Complexity vs. performance
  - Decoding algorithm.
  - Number of iterations.
  - Encoder constraint length
- Latency vs. performance
  - Frame size.
- Spectral efficiency vs. performance
  - Overall code rate
- Other factors
  - Interleaver design.
  - Puncture pattern.
  - Trellis termination.

Performance Bounds for Linear Block Codes

- Union bound for soft-decision decoding:

$$P_b \le \sum_{i=1}^{2^N} \frac{w_i}{N} Q\left(\sqrt{d_i \frac{2rE_b}{N_o}}\right)$$

- For convolutional and turbo codes this becomes:

$$P_b \le \sum_{d=d_{free}}^{n(m+N)} \frac{N_d \tilde{w}_d}{N} Q\left(\sqrt{d \frac{2rE_b}{N_o}}\right)$$

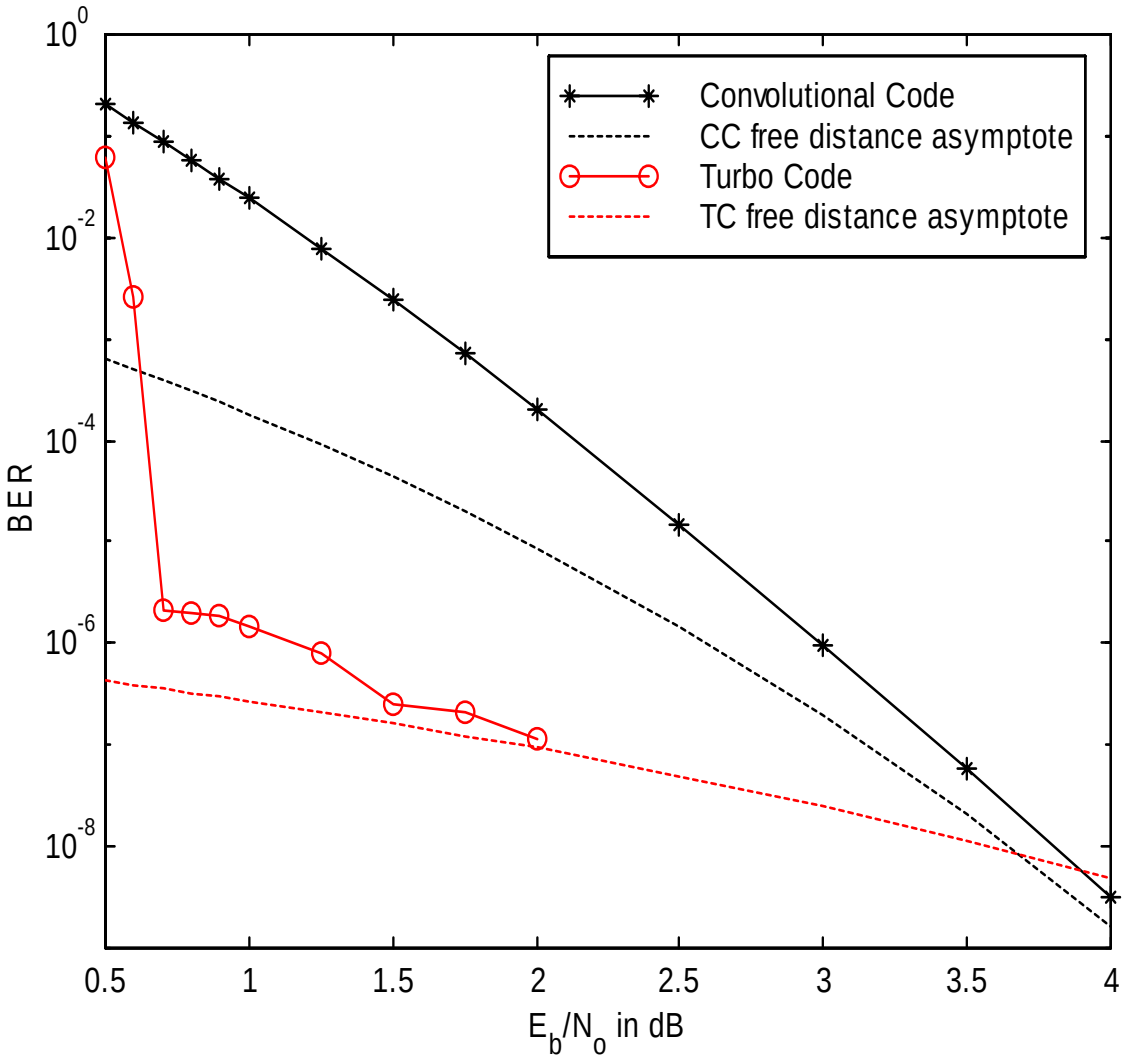
- The free-distance asymptote is the first term of the sum:

$$P_b \approx \frac{N_{free} \tilde{w}_{free}}{N} Q\left(\sqrt{d_{free} \frac{2rE_b}{N_o}}\right)$$

- For convolutional codes N is unbounded and:

$$P_b \approx W_d^0 Q\left(\sqrt{d_{free} \frac{2rE_b}{N_o}}\right)$$

Free-distance Asymptotes



- For convolutional code:
  - $d_{free} = 18$
  - $W_d^o = 187$

$$P_b \approx 187 Q\left(\sqrt{18 \frac{E_b}{N_o}}\right)$$

- For turbo code
  - $d_{free} = 6$
  - $N_{free} = 3$
  - $w_{free} = 2$

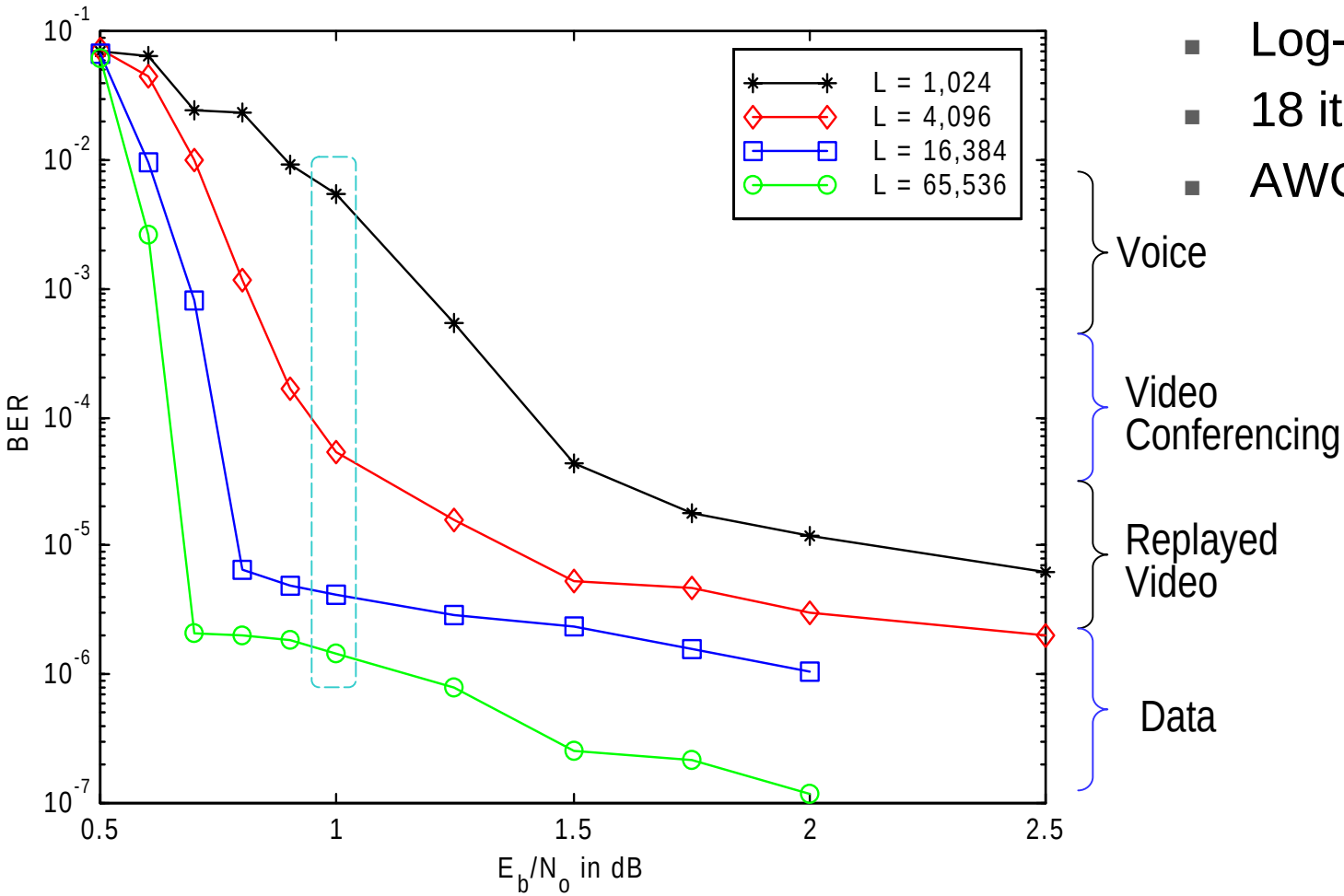
$$P_b \approx \frac{3 \cdot 2}{65536} Q\left(\sqrt{6 \frac{E_b}{N_o}}\right)$$

Application: Turbo Codes for Wireless Multimedia

- Multimedia systems require varying quality of service.
  - QoS
  - Latency
    - Low latency for voice, teleconferencing
  - Bit/frame error rate (BER, FER)
    - Low BER for data transmission.
- The tradeoffs inherent in turbo codes match with the tradeoffs required by multimedia systems.
  - Data: use large frame sizes
    - Low BER, but long latency
  - Voice: use small frame sizes
    - Short latency, but higher BER

Influence of Interleaver Size

- Constraint Length 5.
- Rate r = 1/2.
- Log-MAP decoding.
- 18 iterations.
- AWGN Channel.



Application: Turbo Codes for Fading Channels

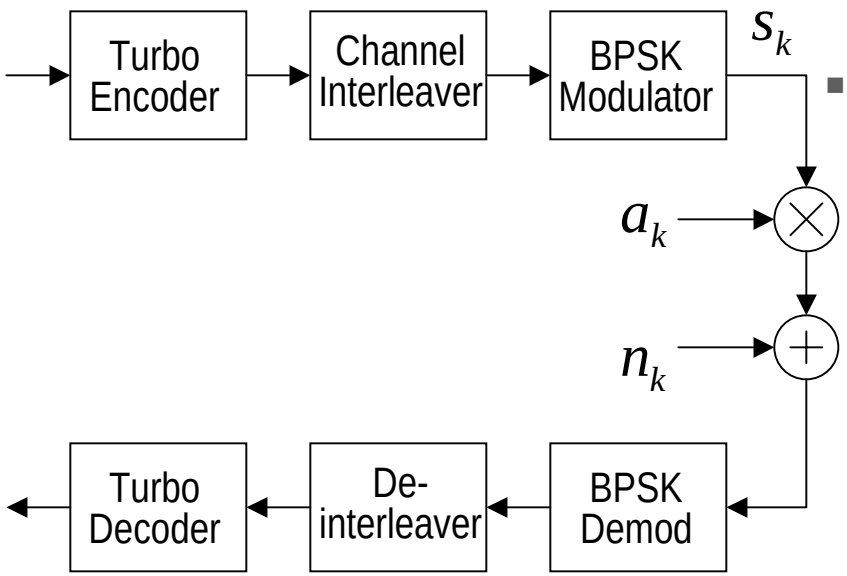
- The turbo decoding algorithm requires accurate estimates of channel parameters:
  - Branch metric:

$$\gamma(s_i \rightarrow s_{i+1}) = \ln P[m_i] + z_i^s x_i^s + z_i^p x_i^p$$

$$z_i = \left(\frac{4 a_i^* E_s}{N_o} \right) r_i = \frac{2}{\sigma^2} r_i a_i^*$$

  - Average signal-to-noise ratio (SNR).
  - Fading amplitude.
  - Phase.
- Because turbo codes operate at low SNR, conventional methods for channel estimation often fail.
  - Therefore channel estimation and tracking is a critical issue with turbo codes.

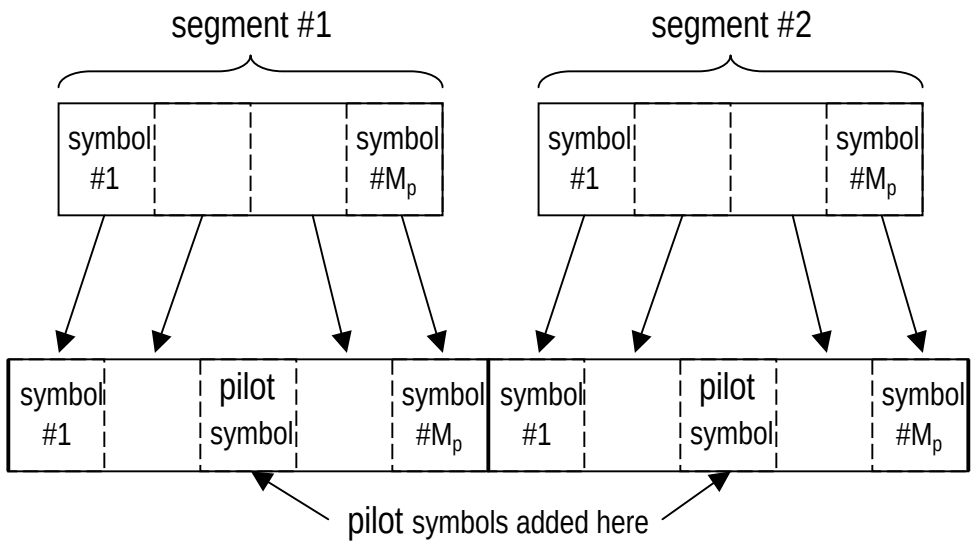
Fading Channel Model



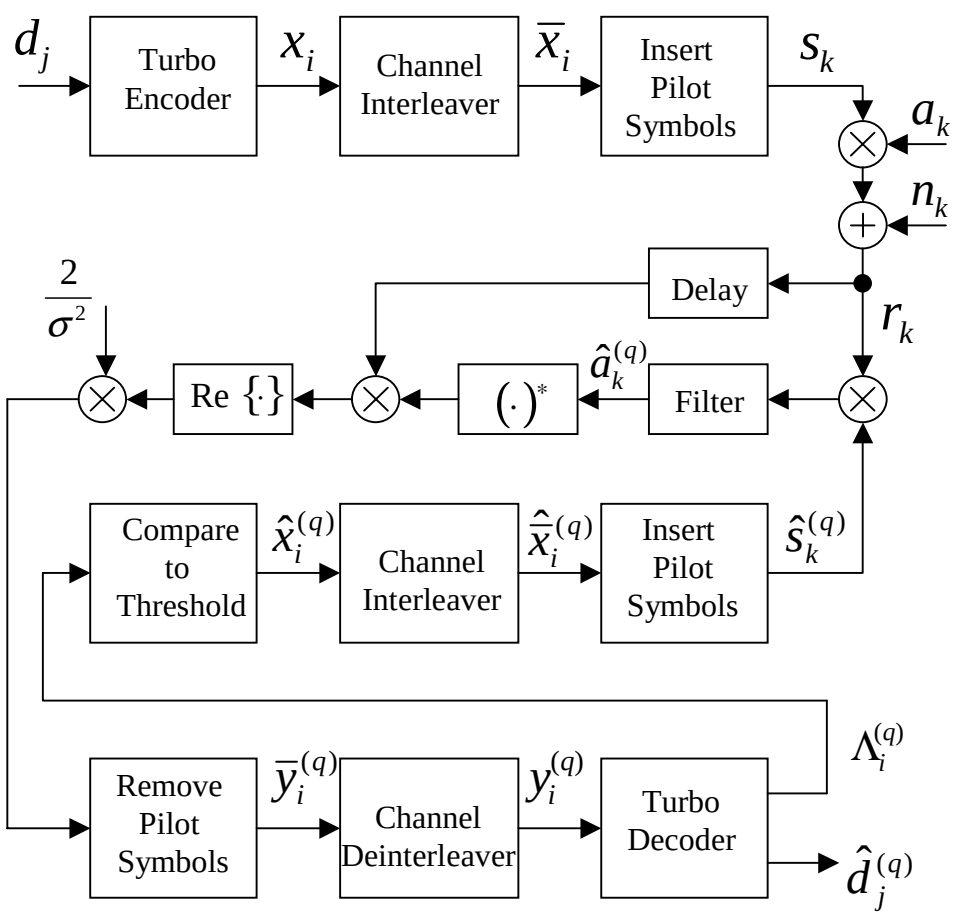
- Antipodal modulation:
$$s_k = \{-1, +1\}$$
- Gaussian Noise:
$$P_n = \frac{N_o}{2E_s}$$
- Complex Fading:
$$a_k = (\alpha + X_k) + jY_k$$
  - α is a constant.
    - α=0 for Rayleigh Fading
    - α>0 for Rician Fading
  - X and Y are Gaussian random processes with autocorrelation:
$$R(k) = J_o(2\pi f_d T_s k)$$

Pilot Symbol Assisted Modulation

- ***Pilot symbols***:
  - Known values that are periodically inserted into the transmitted code stream.
  - Used to assist the operation of a channel estimator at the receiver.
  - Allow for coherent detection over channels that are unknown and time varying.



Pilot Symbol Assisted Turbo Decoding

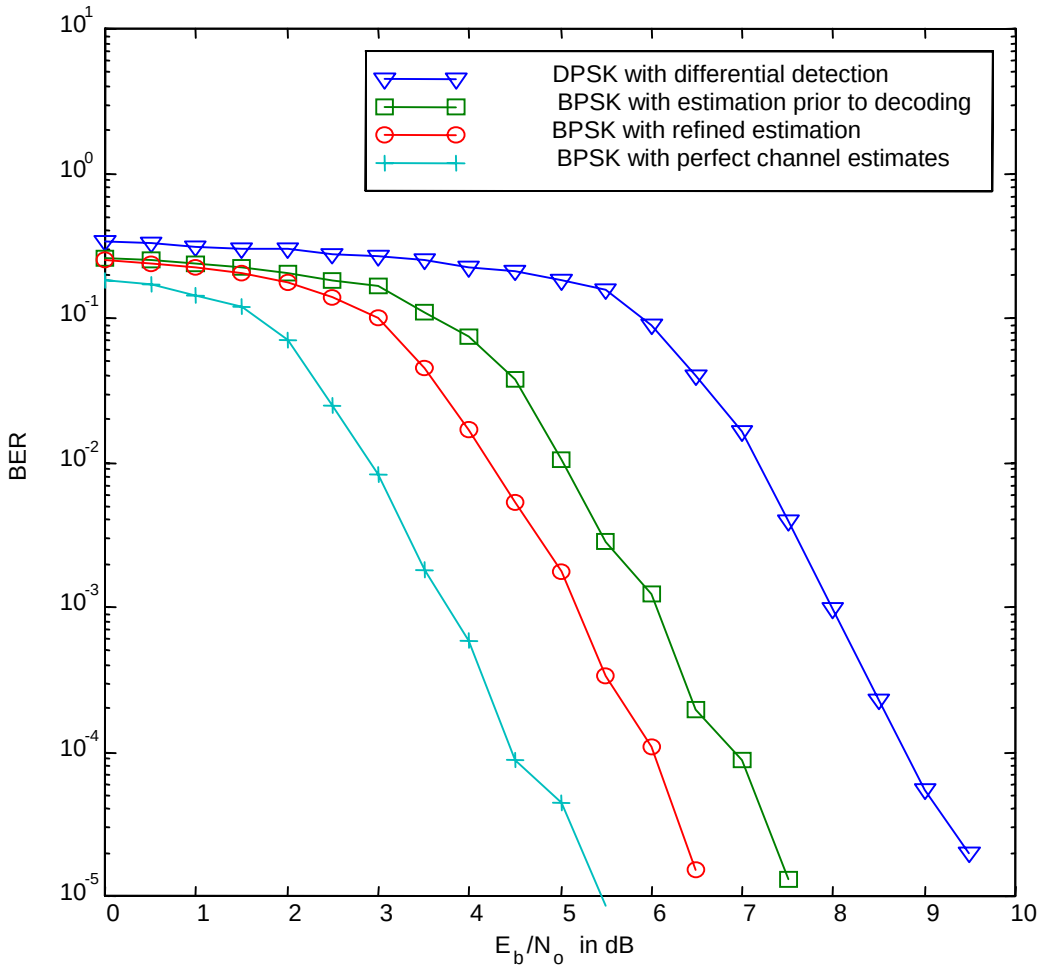


- Desired statistic:

$$\mathrm{Re}\left\{\frac{2}{\sigma^2} r_k a_k^*\right\}$$

- Initial estimates are found using pilot symbols only.
- Estimates for later iterations also use data decoded with high reliability.
- “Decision directed”

Performance of Pilot Symbol Assisted Decoding



- Simulation parameters:
  - Rayleigh flat-fading.
  - r=1/2, K=3
  - 1,024 bit random interleaver.
  - 8 iterations of log-MAP.
  - $f_d T_s = .005$
  - $M_p = 16$
- Estimation prior to decoding degrades performance by 2.5 dB.
- Estimation during decoding only degrades performance by 1.5 dB.
- Noncoherent reception degrades performance by 5 dB.

Other Applications of Turbo Decoding

- The turbo-principle is more general than merely its application to the decoding of turbo codes.
- The “Turbo Principle” can be described as:
  - “Never discard information prematurely that may be useful in making a decision until all decisions related to that information have been completed.”

    -Andrew Viterbi
  - “It is a capital mistake to theorize before you have all the evidence. It biases the judgement.”

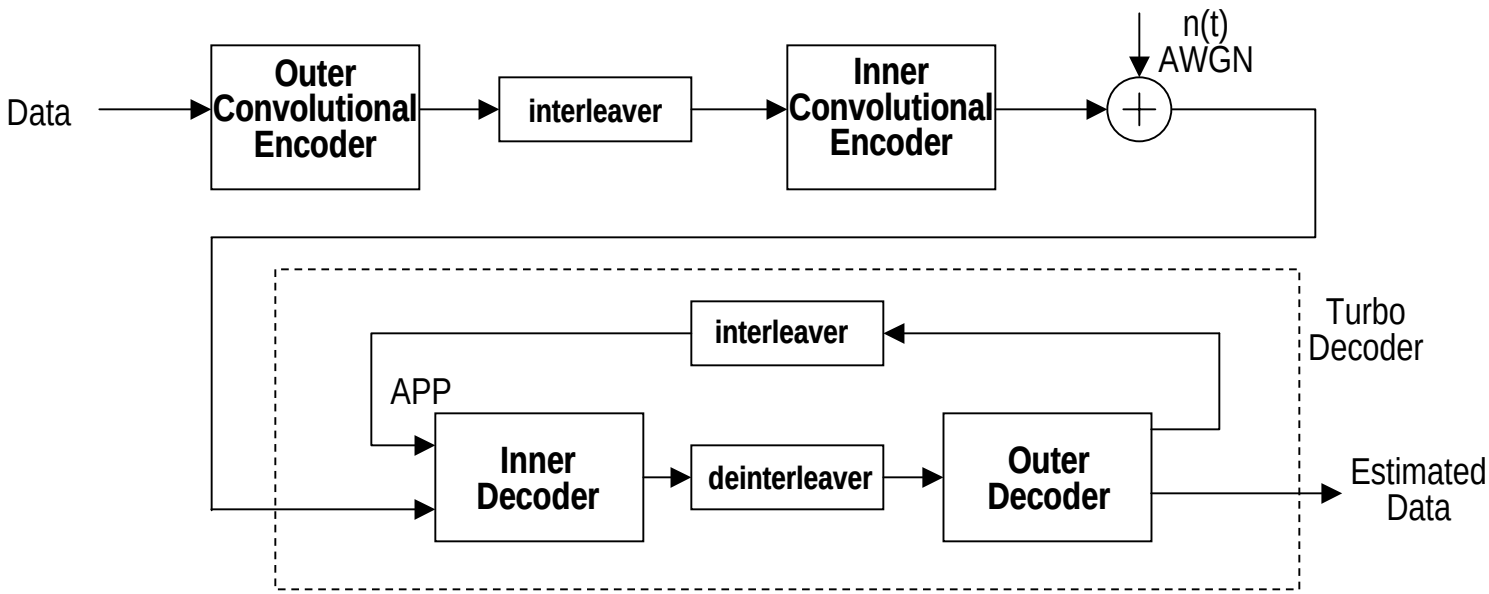
    -Sir Arthur Conan Doyle
- Can be used to improve the interface in systems that employ multiple trellis-based algorithms.

Applications of the Turbo Principle

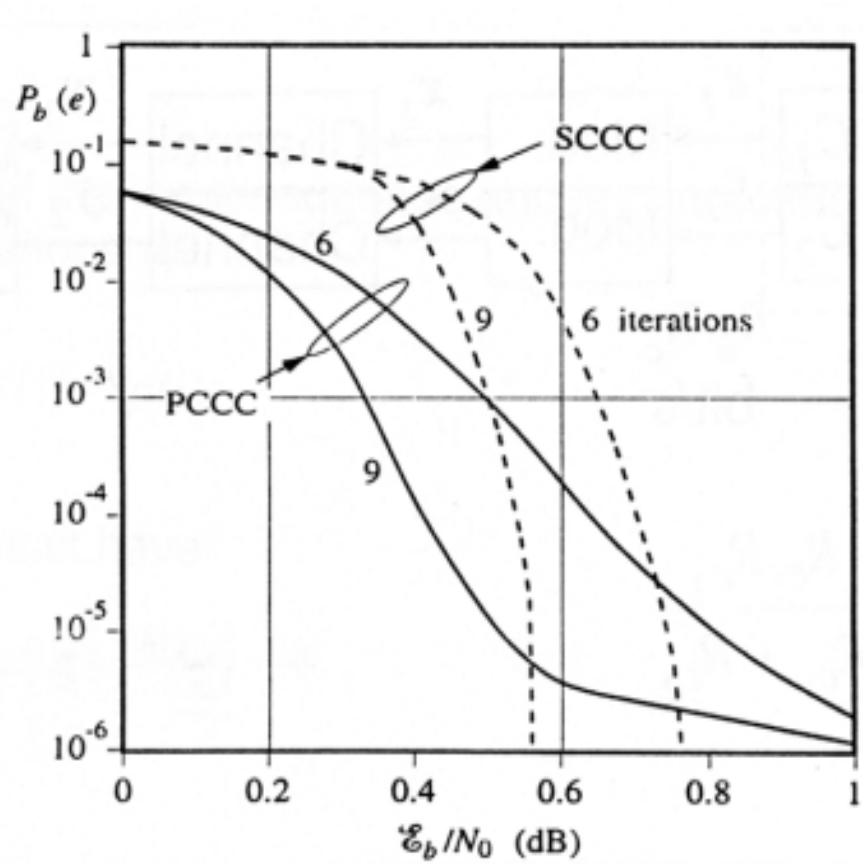
- Other applications of the turbo principle include:
  - Decoding serially concatenated codes.
  - Combined equalization and error correction decoding.
  - Combined multiuser detection and error correction decoding.
  - (Spatial) diversity combining for coded systems in the presence of MAI or ISI.

Serial Concatenated Codes

- The turbo decoder can also be used to decode serially concatenated codes.
  - Typically two convolutional codes.



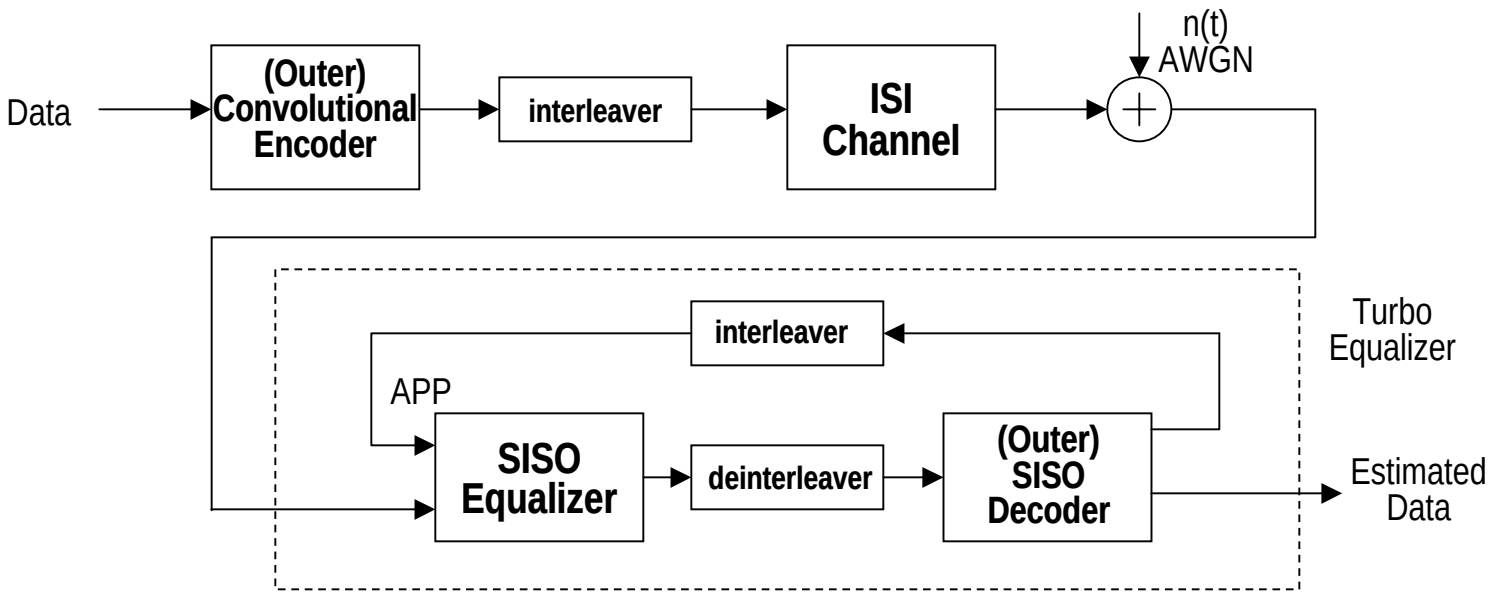
Performance of Serial Concatenated Turbo Code



- Plot is from:
  S. Benedetto, et al "Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding" Proc., *Int. Symp. on Info. Theory*, 1997.
- Rate r=1/3.
- Interleaver size L = 16,384.
- K = 3 encoders.
- Serial concatenated codes do not seem to have a bit error rate floor.

Turbo Equalization

- The “inner code” of a serial concatenation could be an Intersymbol Interference (ISI) channel.
  - ISI channel can be interpreted as a rate 1 code defined over the field of real numbers.



Performance of Turbo Equalizer

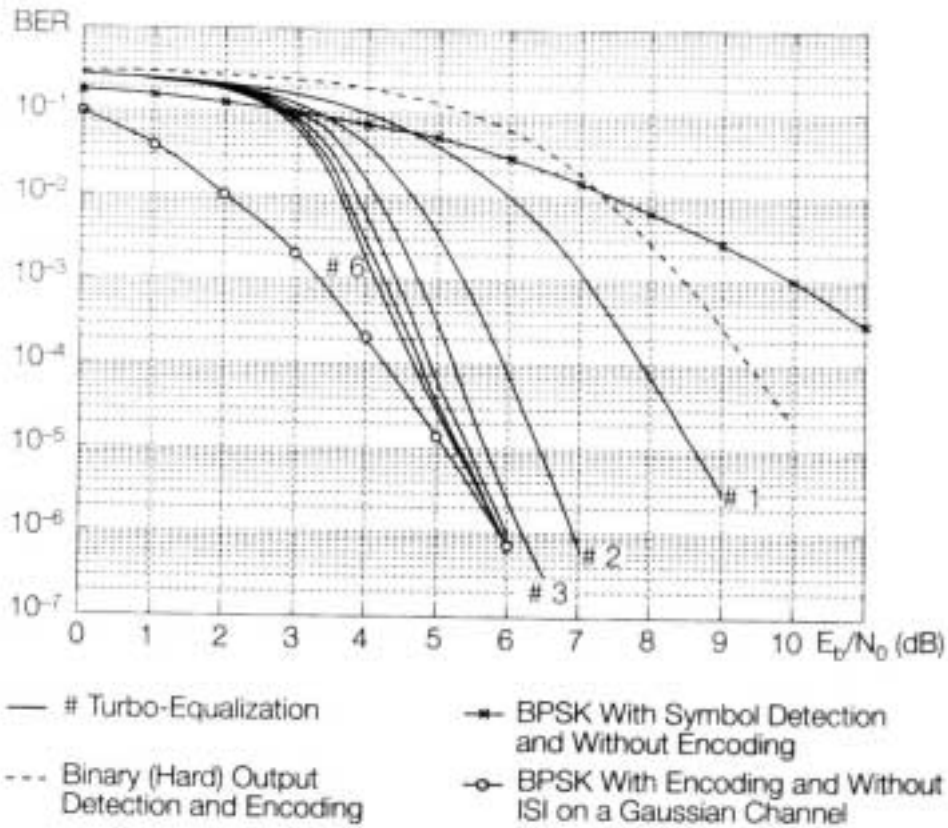


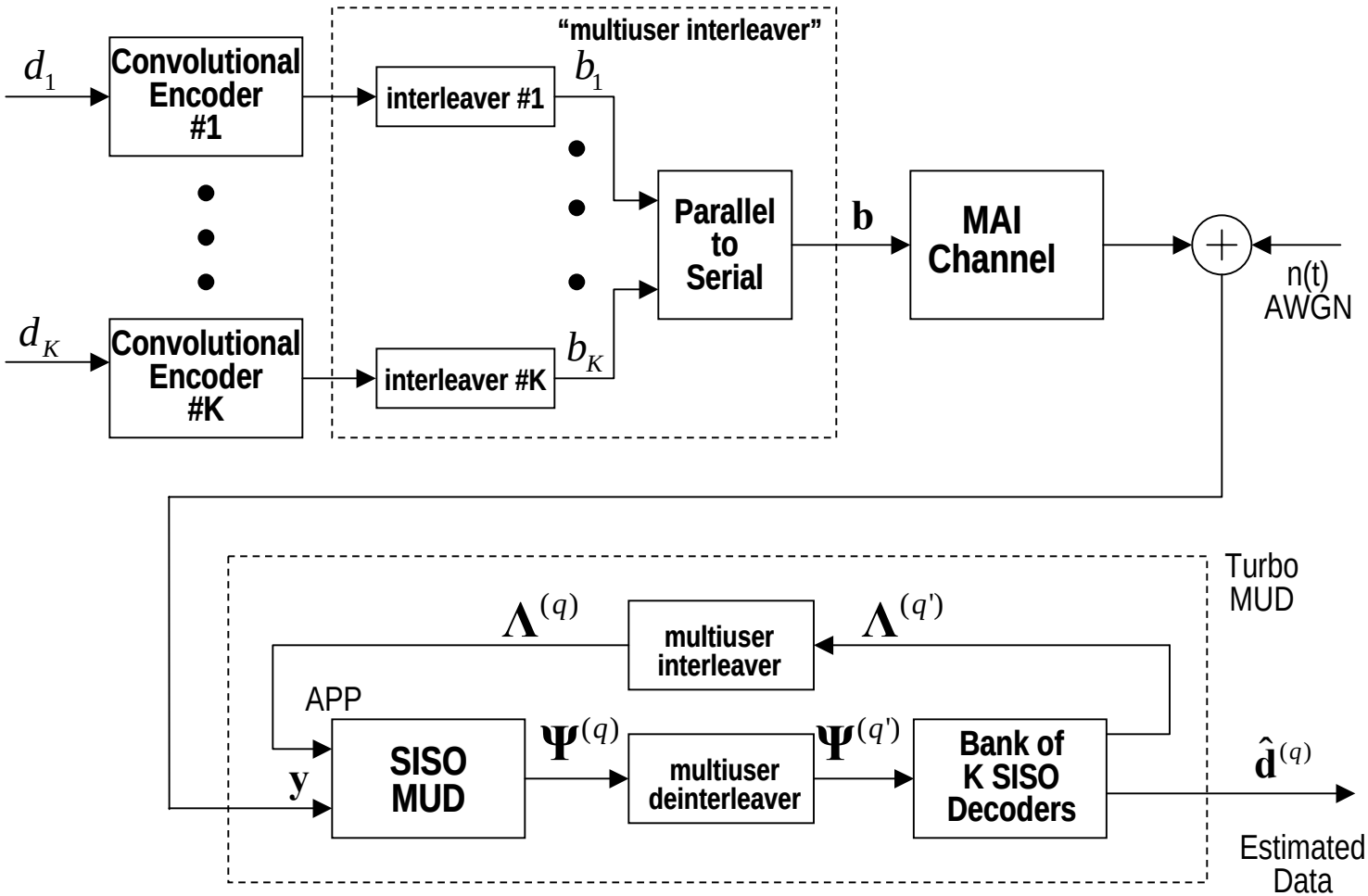
Fig. 6 - Performance of turbo-equalization over a gaussian channel (convolutional encoding with $K = 5$).

- Plot is from:
  C. Douillard,et al "Iterative Correction of Intersymbol Interference: Turbo-Equaliztion", *European Transactions on Telecommuications*, Sept./Oct. 1997.
- M=5 independent multipaths.
  - Symbol spaced paths
  - Stationary channel.
  - Perfectly known channel.
- (2,1,5) convolutional code.

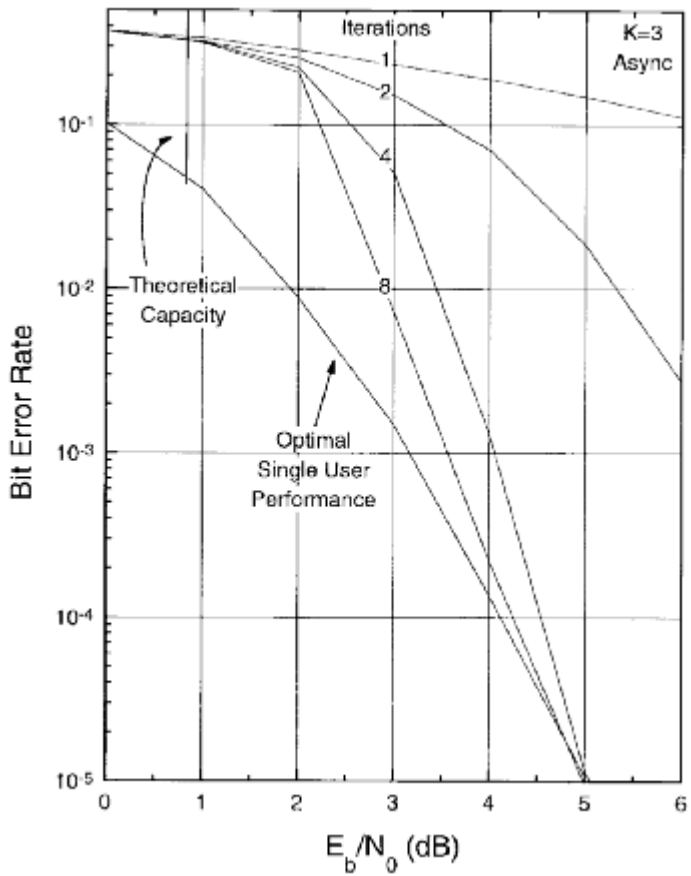
Turbo Multiuser Detection

- The “inner code” of a serial concatenation could be a multiple-access interference (MAI) channel.
  - MAI channel describes the interaction between K nonorthogonal users sharing the same channel.
  - MAI channel can be thought of as a time varying ISI channel.
  - MAI channel is a rate 1 code with time-varying coefficients over the field of real numbers.
  - The input to the MAI channel consists of the encoded and interleaved sequences of all K users in the system.
- MAI channel can be:
  - CDMA: Code Division Multiple Access
  - TDMA: Time Division Multiple Access

System Diagram



Simulation Results: MAI Channel w/ AWGN



- From:
  - M. Moher, “An iterative algorithm for asynchronous coded multiuser detection,” IEEE Comm. Letters, Aug.1998.
- Generic MA system
  - K=3 asynchronous users.
  - Identical pulse shapes.
  - Each user has its own interleaver.
- Convolutionally coded.
  - Constraint length 3.
  - Code rate 1/2.
- Iterative decoder.

Conclusion

- Turbo code advantages:
  - Remarkable power efficiency in AWGN and flat-fading channels for moderately low BER.
  - Deign tradeoffs suitable for delivery of multimedia services.
- Turbo code disadvantages:
  - Long latency.
  - Poor performance at very low BER.
  - Because turbo codes operate at very low SNR, channel estimation and tracking is a critical issue.
- The principle of iterative or “turbo” processing can be applied to other problems.
  - Turbo-multiuser detection can improve performance of coded multiple-access systems.