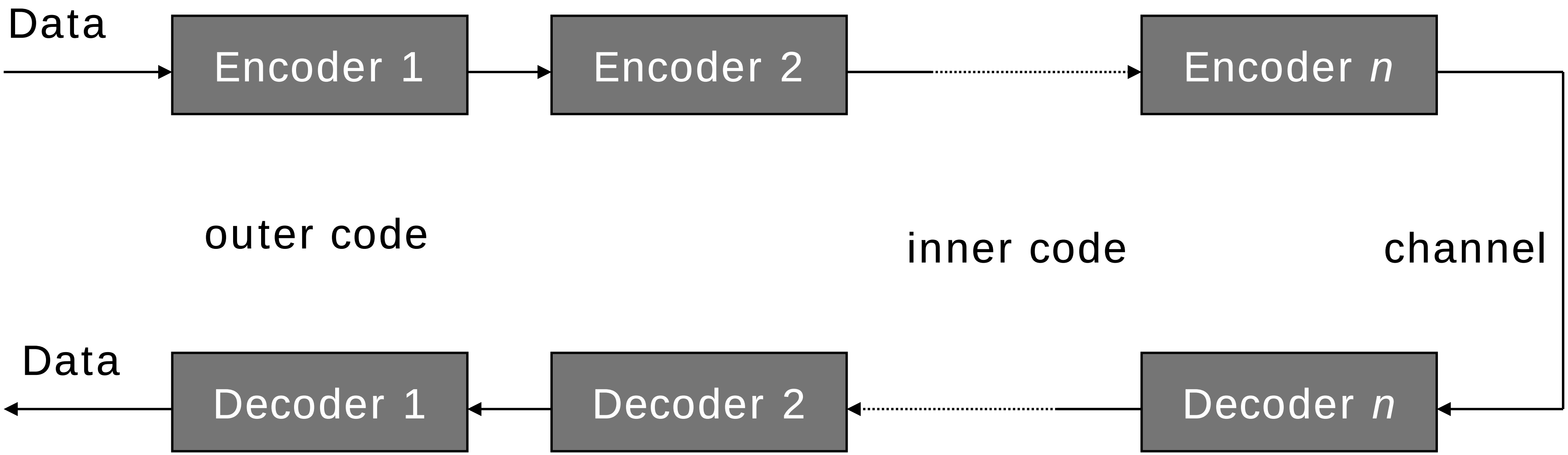
Principle of Turbo Codes

Concatenated Codes

- The power of forward error correcting (FEC) codes increases with length *k*.
- Decoding complexity also increases very rapidly with length *k*.
- Solve the problem by building a long, complex code out of much shorter component codes, which can be decoded much more easily.
- This is called *concatenation* technique.

Principle of Concatenated Codes

- Serial concatenation



Principle of Concatenated Codes (Cont'd)

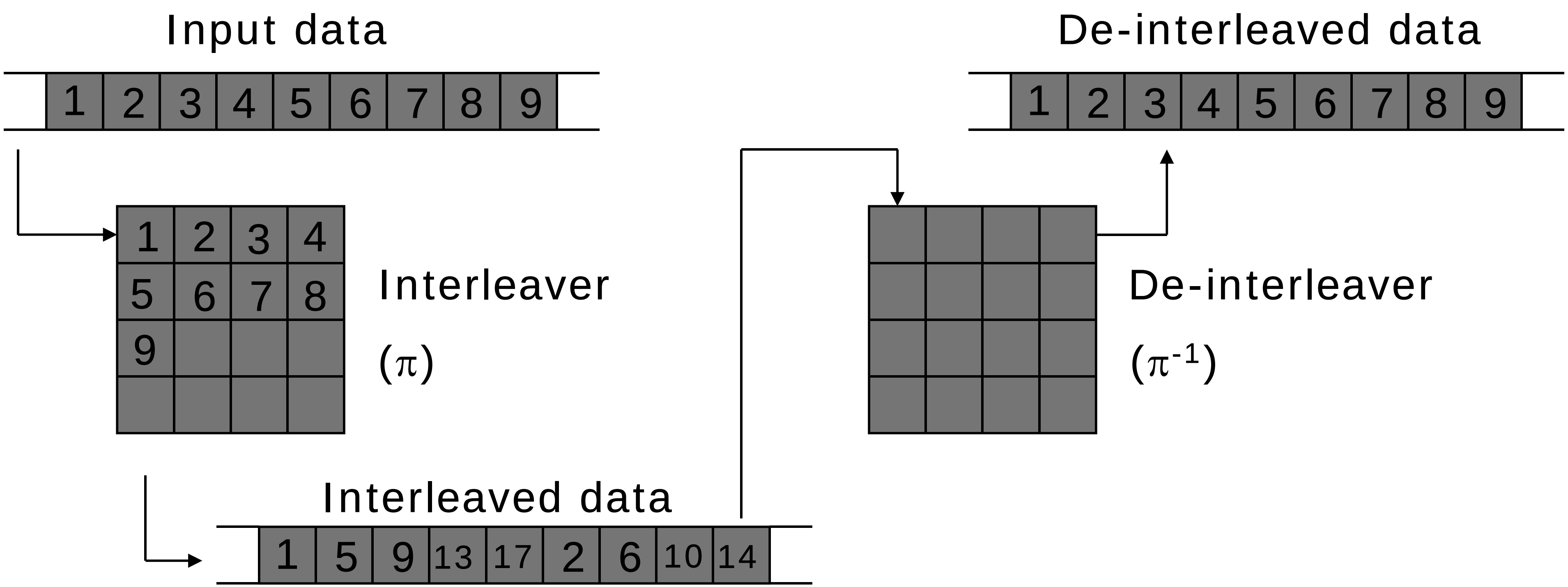
- Most significant drawback: error propagation
- Decoding error in a codeword of one decoder will be passed to the next decoder
- Too many errors in one codeword may overwhelm the decoder to correct the error

Principle of Concatenated Codes (Cont'd)

- Improve performance by distributing these errors to a number of separate codewords before inputting into the next decoder
- This is achieved using *interleaver* and *de-interleaver*

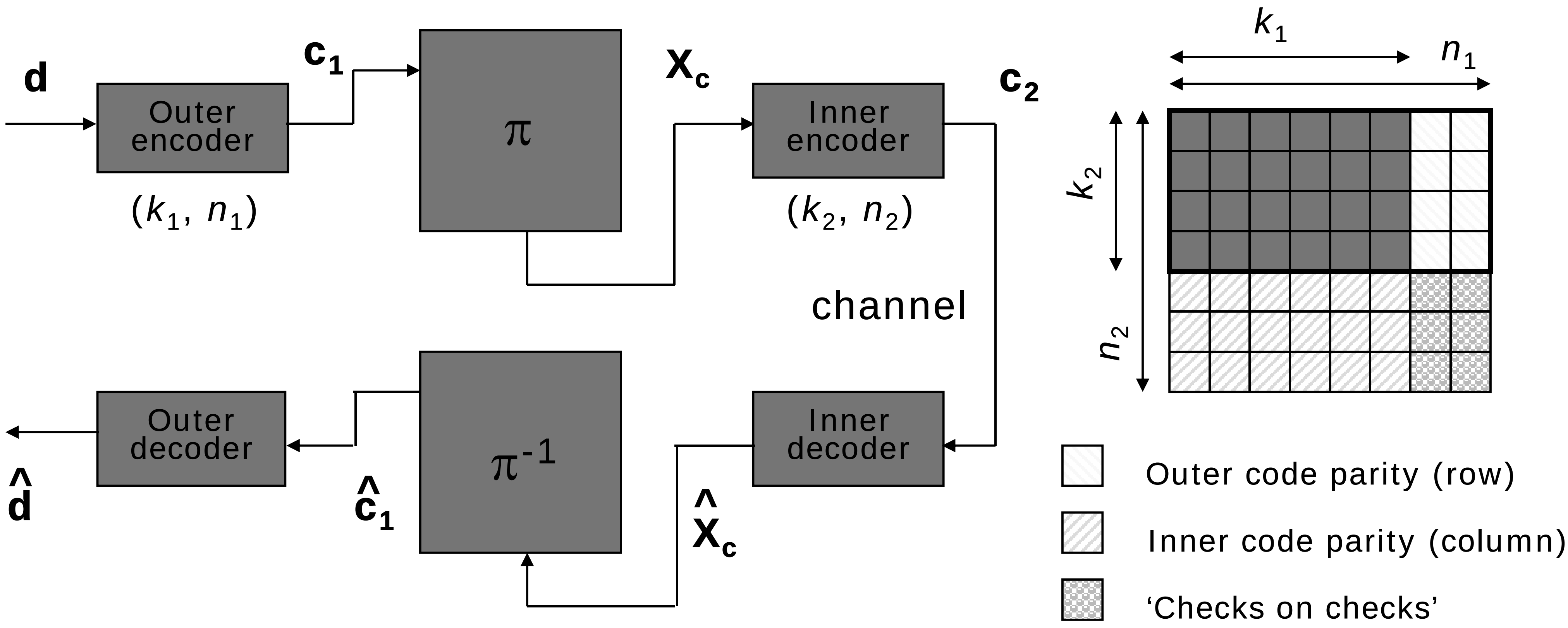
Principle of Concatenated Codes (Cont'd)

- Block or rectangular interleaver



Principle of Concatenated Codes (Cont'd)

- Concatenated code with interleaver



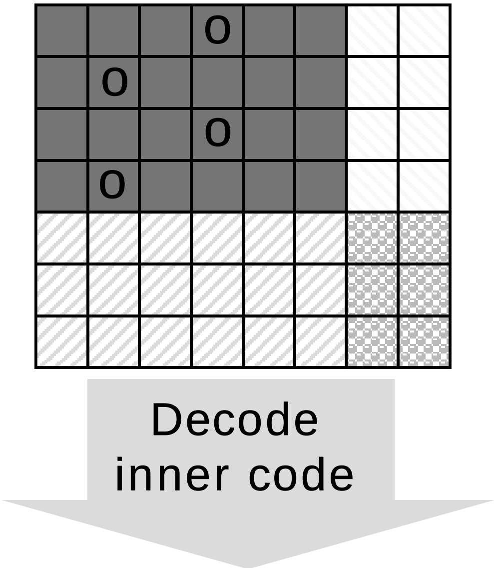
Principle of Concatenated Codes (Cont'd)

- Usually the block codes used in concatenated coding scheme are systematic
- Array within the heavy line box is stored in the interleaver array (k_2 x n_1 dimension)
- The composite code is much longer and more powerful
- The data length is k_1 x k_2 and overall length is n_1 x n_2
- This is called *array* or *product code*

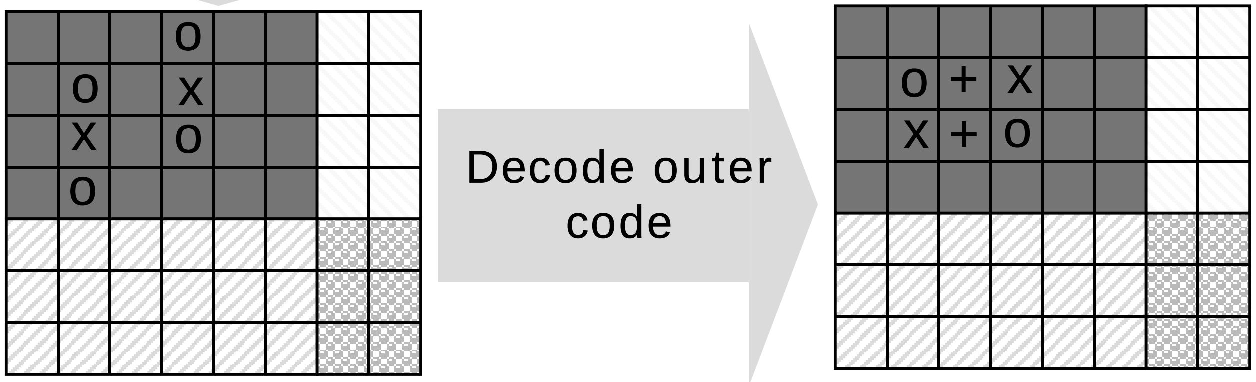
Principle of Concatenated Codes (Cont'd)

- Conventional decoding technique: decode inner code, then the outer
- It may not always be as effective as we might hope
- Assume both component codes are capable of correcting single errors only
- The 'O's are original received error patterns

Principle of Concatenated Codes (Cont’d)



‘x’ indicates new errors added by inner decoder



- ‘+’ indicates new errors added by outer decoder
- Some original errors (‘o’) are corrected

Principle of Concatenated Codes (Cont'd)

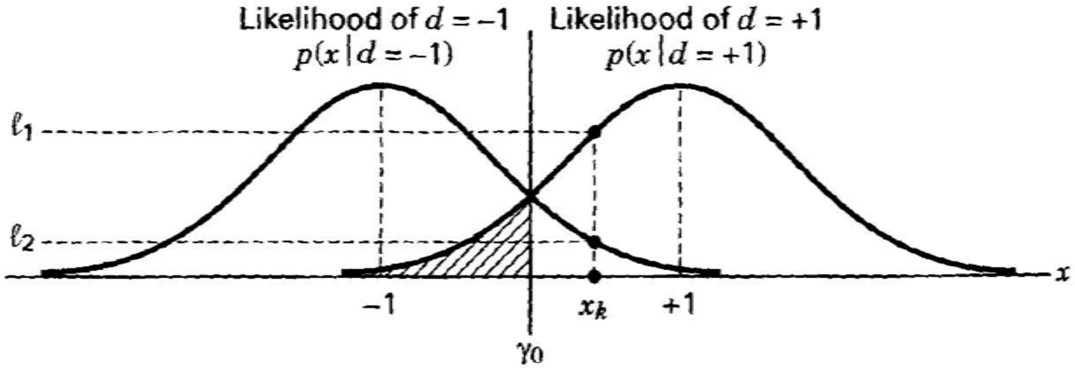
- The final decoded codewords contained more errors than the original received codewords
- If the output of the outer decoder are re-applied to the inner decoder it will detect that some errors remained
- This is the *principle of iterative decoder* or *'turbo' principle*

Principle of Concatenated Codes (Cont'd)

- However, for the above case the inner decoder may not be able to correct the errors
- The inner decoder needs additional information
- Hard decision made by demodulator and decoders destroy this information
- Needs soft-in, soft-out (SISO) decoding

Turbo decoding

- Maximum a posteriori (MAP) decoding
- At the demodulator output:



- Hypothesis H_1: likelihood is $d = +1$
- Hypothesis H_2: likelihood is $d = -1$

Turbo Decoding (Cont'd)

- At time k:
  - Demodulator output = x_k
  - $\ell_1 > \ell_2$, or $P(x_k|d=+1) > \mathrm{P}(x_k|d=-1)$
- Decision Rule:

$$P(d=+1\,|\,x) \underset{H_2}{\overset{H_1}{\gtrless}} P(d=-1\,|\,x)$$

Turbo Decoding (Cont'd)

- Using Bayes' theorem:

$$P(d=+1 \mid x)p(x) \underset{H_2}{\overset{H_1}{\gtrless}} P(d=-1 \mid x)p(x)$$

$$p(x \mid d=+1)P(d=+1) \underset{H_2}{\overset{H_1}{\gtrless}} p(x \mid d=-1)P(d=-1)$$

Turbo Decoding (Cont'd)

- Express in terms of a ratio:

$$\frac{P(d=+1\,|\,x)}{P(d=-1\,|\,x)} \underset{H_2}{\overset{H_1}{\gtrless}} 1 \quad or \quad \frac{p(x\,|\,d=+1)P(d=+1)}{p(x\,|\,d=-1)P(d=-1)} \underset{H_2}{\overset{H_1}{\gtrless}} 1$$

- Take logarithm of LHS, we have Log-Likelihood Ratio (LLR):

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

Turbo Decoding (Cont'd)

- Hence,

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

- $L(x|d)$ is the LLR of the test statistic x obtained by measurement of x under the conditions d=+1 or d=-1.
- $L(d)$ is the a priori LLR of the data bit d.

Turbo Decoding (Cont'd)

- At the decoder output:
- For systematic code, it has been shown that the LLR (soft decoder output) is

$$L_d(d|x) = L(d|x) + L_e(d|x)$$

- Where $L_e(d|x)$ = extrinsic LLR, represents extra knowledge that is gleaned from the decoding process.
- Hence,

$$L_d(d|x) = L(x|d) + L(d) + L_e(d|x)$$

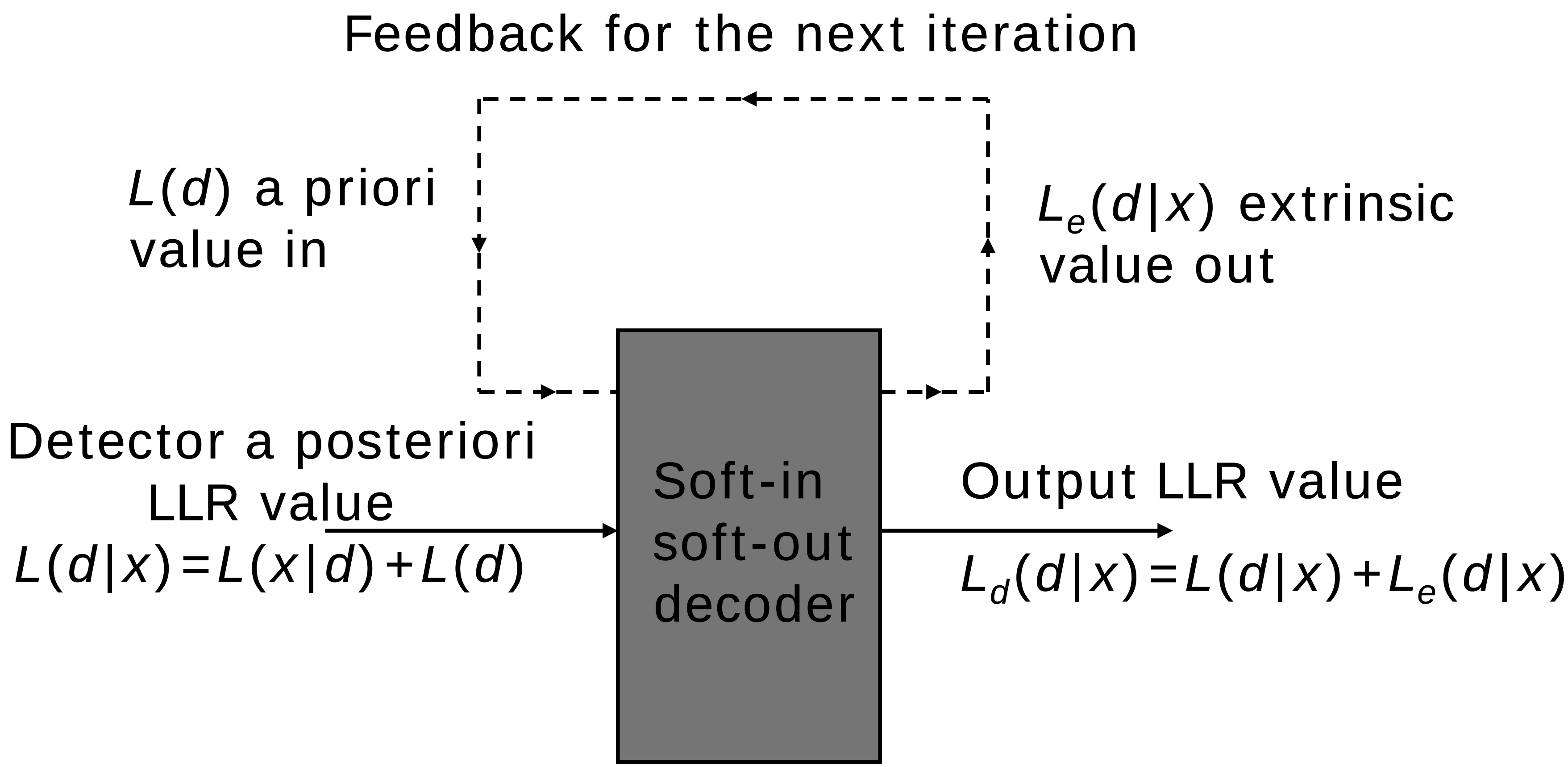
Turbo Decoding (Cont'd)

- The sign of $L_d(d|x)$ denotes hard decision:
  - Positive: $d = +1$
  - Negative: $d = -1$
- The magnitude of $L_d(d|x)$ denotes the reliability of that decision.

Turbo Decoding (Cont'd)

- Iterative (turbo) decoding:
  - Assume initially the binary data to be equally likely: set $L(d) = 0$
  - Measure x and calculate ℓ_1 and ℓ_2, then calculate $L(x|d)$, hence $L(d|x)$
  - Decode and obtain $L_e(d|x)$ from decoder output
  - Feedback $L_e(d|x)$ and set $L(d) = L_e(d|x)$
  - Re-calculate $L(d|x)$ and decode

Turbo Decoding (Cont’d)



Turbo Decoding (Example)

- Encoder output binary digits:

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

Note:

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

- Demodulator output (due to noise):

$x_1=0.75$	$x_2=0.05$	$x_{12}=1.25$
$x_3=0.10$	$x_4=0.15$	$x_{34}=1.00$
$x_{13}=3.00$	$x_{24}=0.50$	

Assume the following conversion:
$0 \rightarrow -1$
$1 \rightarrow +1$

Turbo Decoding (Example)

- Decoder input log-likelihood ratio, $L(x|d)+0$:

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

$$= \ln\left(\frac{\dfrac{1}{\sigma\sqrt{2\pi}}\exp\left[-\dfrac{1}{2}\left(\dfrac{x-1}{\sigma}\right)^2\right]}{\dfrac{1}{\sigma\sqrt{2\pi}}\exp\left[-\dfrac{1}{2}\left(\dfrac{x+1}{\sigma}\right)^2\right]}\right)$$

$$= -\frac{1}{2}\left(\frac{x-1}{\sigma}\right)^2 + \frac{1}{2}\left(\frac{x+1}{\sigma}\right)^2 = \frac{2}{\sigma^2}x$$

Turbo Decoding (Example)

- To simplify, assume $\sigma^2 = 1$, then

$$L(x|d) = L(x) = 2x$$

- Hence,

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

- If hard decisions are made without decoding, d_2 and d_3 will be in errors.

Turbo Decoding (Example)

- LLR of modulo-2 sum of two bits:

$$L(b_1 \oplus b_2) = \ln\left[\frac{e^{L(b_1)} + e^{L(b_2)}}{1 + e^{L(b_1)}e^{L(b_2)}}\right]$$

$$\approx (-1) \times \operatorname{sgn}\left[L(b_1)\right] \times \operatorname{sgn}\left[L(b_2)\right]$$

$$\times \min\left(\left|L(b_1)\right|, \left|L(b_2)\right|\right)$$

Turbo Decoding (Example)

- Extrinsic LLR calculation [to simplify notation, $L_e(d_i|x_i) = L_e(\hat{d}_i)$]:

$$
\begin{aligned}
L_e\left(\hat{d}_i\right) &= L\left(\hat{d}_j \oplus \hat{p}_{ij}\right) \\
&\approx (-1)\cdot \operatorname{sgn}\left[L(\hat{d}_j)\right]\cdot \operatorname{sgn}\left[L(\hat{p}_{ij})\right] \\
&\quad \cdot \min\left(\left|L(\hat{d}_j)\right|, \left|L(\hat{p}_{ij})\right|\right) \\
&= (-1)\cdot \operatorname{sgn}\left[L(x_j)+L(d_j)\right]\cdot \operatorname{sgn}\left[L(x_{ij})\right] \\
&\quad \cdot \min\left(\left|L(x_j)+L(d_j)\right|, \left|L(x_{ij})\right|\right)
\end{aligned}
$$

Turbo Decoding (Example)

- Assume $L(\hat{p}_{ij}) = L(x_{ij})$ because p_{ij} depends on d_i and d_j.
- Decoding horizontally:

$$
\begin{aligned}
L_{eh}(\hat{d}_1) &= (-1)\cdot \operatorname{sgn}\left[L(x_2)+L(d_2)\right]\cdot \operatorname{sgn}\left[L(x_{12})\right] \\
&\quad \cdot \min\left(\left|L(x_2)+L(d_2)\right|, \left|L(x_{12})\right|\right) \\
&= (-1)\cdot \operatorname{sgn}\left[0.1+0\right]\cdot \operatorname{sgn}\left[2.5\right]\cdot (0.1) \\
&= (-1)\cdot(+1)\cdot(+1)\cdot(0.1) \\
&= -0.1 = \mathit{new}\ L(d_1)
\end{aligned}
$$

Turbo Decoding (Example)

- Similarly:
  - $L_{eh}(\hat{d}_2) = -\ \text{sgn}(1.5+0)\cdot\text{sgn}(2.5)\cdot(1.5)$

    $= -\ 1.5 = \text{new } L(d_2)$
  - $L_{eh}(\hat{d}_3) = -\ \text{sgn}(0.3+0)\cdot\text{sgn}(2.0)\cdot(0.3)$

    $= -\ 0.3 = \text{new } L(d_3)$
  - $L_{eh}(\hat{d}4) = -\ \text{sgn}(0.2+0)\cdot\text{sgn}(2.0)\cdot(0.2)$

    $= -\ 0.2 = \text{new } L(d_4)$

Turbo Decoding (Example)

- Decoding vertically:
  - $L_{ev}(\hat{d}_1) = 0.1 = \text{new } L(d_1)$
  - $L_{ev}(\hat{d}_2) = -0.1 = \text{new } L(d_2)$
  - $L_{ev}(\hat{d}_3) = -1.4 = \text{new } L(d_3)$
  - $L_{ev}(\hat{d}_4) = 1.0 = \text{new } L(d_4)$
- Final decoder output log-likelihood ratio after first iteration:

$$L_d(d_i|x_i) = L(x_i) + L_{eh}(\hat{d}_i) + L_{ev}(\hat{d}_i)$$

Turbo Decoding (Example)

- Hence,
  - Final $L_d(d_1|x_1)$ = 1.5 - 0.1 + 0.1 = 1.5
  - Final $L_d(d_2|x_2)$ = 0.1 - 1.5 - 0.1 = -1.5
  - Final $L_d(d_3|x_3)$ = 0.2 - 0.3 - 1.4 = -1.5
  - Final $L_d(d_4|x_4)$ = 0.3 - 0.2 + 1.0 = 1.1
- Final decoder output after 1st iteration:

$L_d(d_1\|x_1)=1.5$	$L_d(d_2\|x_2)= -1.5$
$L_d(d_3\|x_3)= -1.5$	$L_d(d_4\|x_4)=1.1$

Turbo Decoding (Example)

- After 1st iteration, it is sufficient to yield correct hard decision outputs for d_3 and d_4.
- Let's see if 2nd iteration can improve the reliability or higher confidence.
- For 2nd iteration, repeat horizontal and vertical decodings with new $L(d_i)$ values.

Turbo Decoding (Example)

- Horizontal decoding:
  - $L_{eh}(\hat{d}_1)$ = -sgn(0.1-0.1)·sgn(2.5)·(0)
    = 0 = new $L(d_1)$
  - $L_{eh}(\hat{d}_2)$ = -sgn(1.5+0.1)·sgn(2.5)·(1.6)
    = - 1.6 = new $L(d_2)$
  - $L_{eh}(\hat{d}_3)$ = -sgn(0.3+1.0)·sgn(2.0)·(1.3)
    = - 1.3 = new $L(d_3)$
  - $L_{eh}(\hat{d}_4)$ = -sgn(0.2-1.4)·sgn(2.0)·(|-1.2|)
    = 1.2 = new $L(d_4)$

Turbo Decoding (Example)

- Vertical decoding:
  - $L_{ev}(d_1) = 1.1 = \text{new } \mathrm{L}(d_1)$
  - $L_{ev}(d_2) = -1.0 = \text{new } \mathrm{L}(d_2)$
  - $L_{ev}(d_3) = -1.5 = \text{new } \mathrm{L}(d_3)$
  - $L_{ev}(d_4) = 1.0 = \text{new } \mathrm{L}(d_4)$
- Final LLR after 2nd iteration:
  - $L_d(d_1|x_1) = 1.5 + 0 + 1.1 = 2.6$
  - $L_d(d_2|x_2) = 0.1 - 1.6 - 1.0 = -2.5$
  - $L_d(d_3|x_3) = 0.2 - 1.3 - 1.5 = -2.6$
  - $L_d(d_4|x_4) = 0.3 + 1.2 + 1.0 = 2.5$

Turbo Decoding (Example)

- Hence final decoder output after 2nd iteration:

$L_d(d_1 \vert x_1) = 2.6$	$L_d(d_2 \vert x_2) = -2.5$
$L_d(d_3 \vert x_3) = -2.6$	$L_d(d_4 \vert x_4) = 2.5$

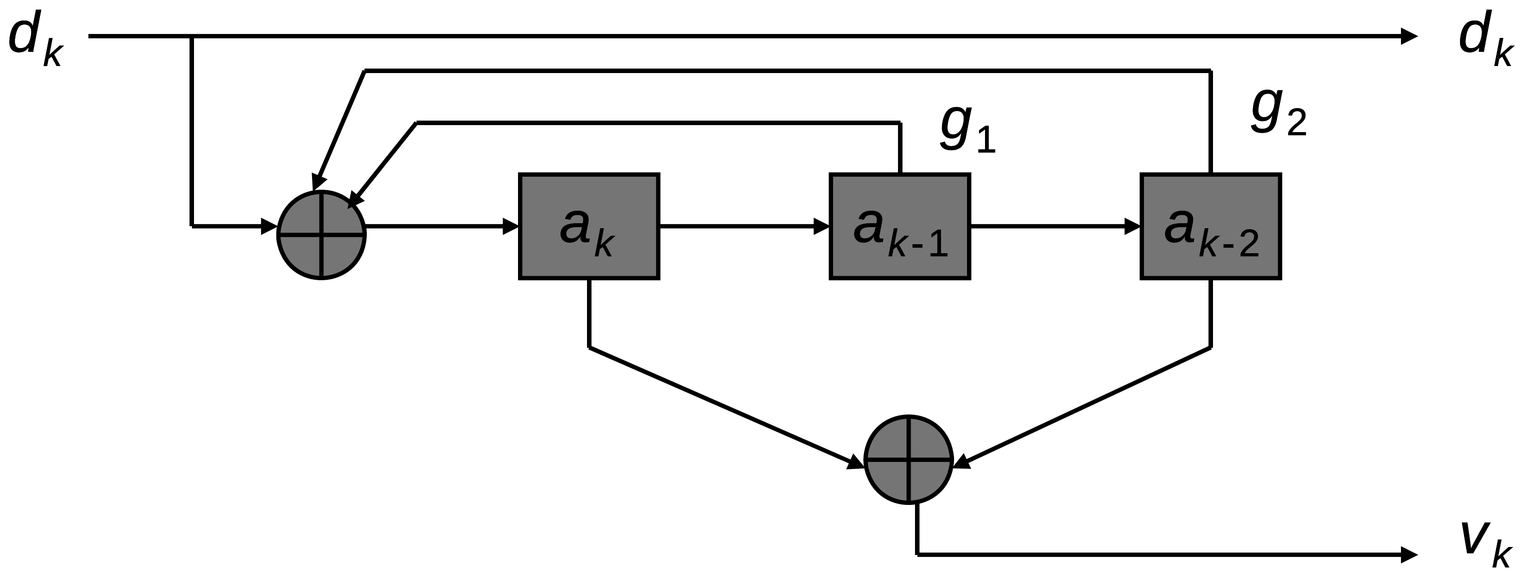
- We see that the level of decision confidence increased.

Turbo-Convolutional Codes

- Invented by C. Berrou, A. Glavieux and P. Thitimajshima in 1993.
- It is the original Turbo Codes.
- It is a parallel (not serial) concatenated recursive systematic convolutional (RSC) codes.
- Recursive codes are used because they give better performance than the best non-systematic codes at all E_b/N_0 for high code rates.

Turbo-Convolutional Codes (Cont'd)

- RSC code (constraint length, K = 3):

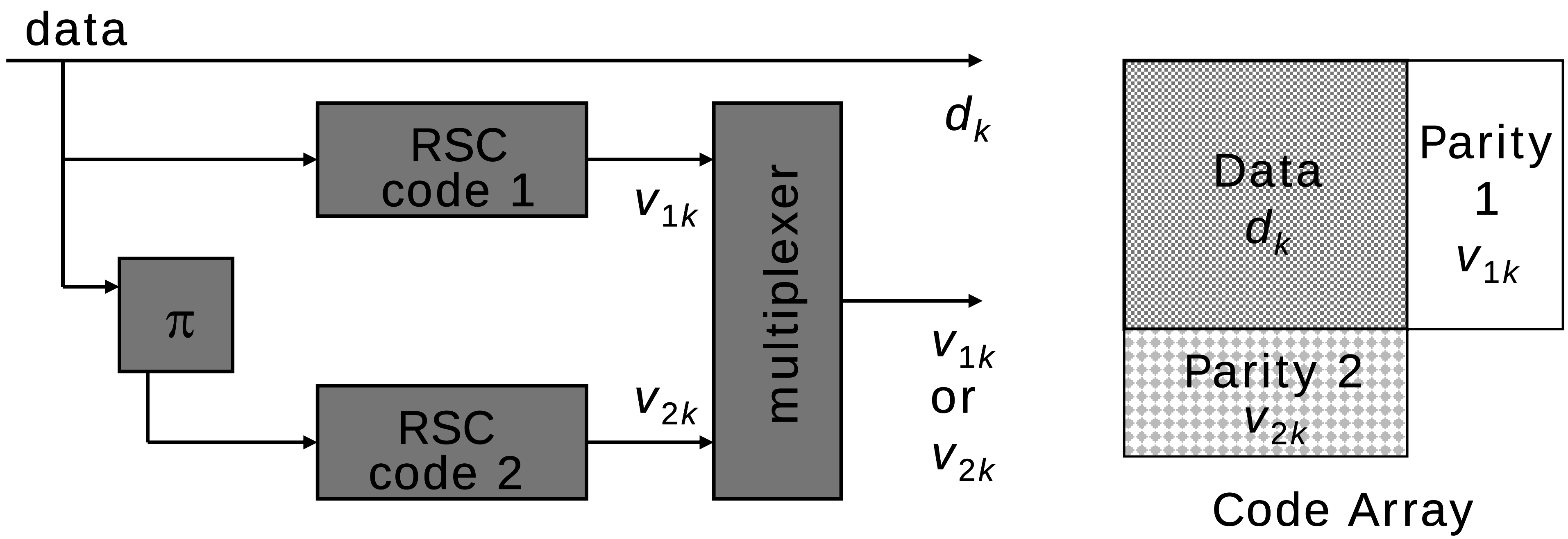


- Mathematical representation:

$$a_k = d_k + \sum_{i=1}^{K-1} g_i a_{k-i} \quad \text{mod-}2$$

Turbo-Convolutional Codes (Cont'd)

- Structure of parallel concatenation:



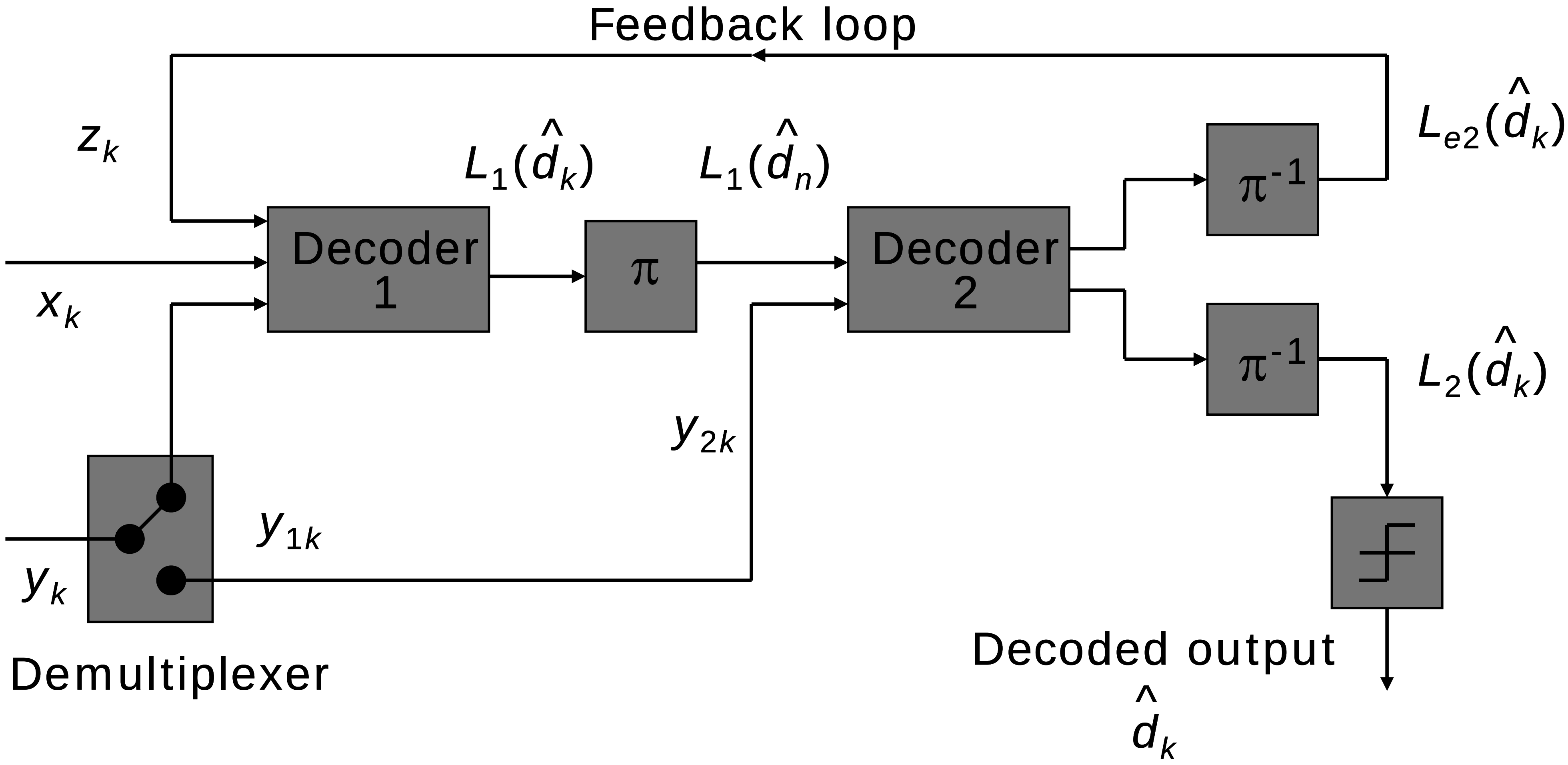
- Code array is the same as serial concatenation, except no 'check-on-check'.

Turbo-Convolutional Codes (Cont'd)

- Iterative decoding:
  - Viterbi Algorithm is an optimal decoding method for minimizing the probability of sequence error, not bit error.
  - Output of Viterbi is hard decision of a sequence of bits.
  - For iterative decoding, we need soft-decision output for each decoded bit.
- Need to use Bahl Algorithm (beyond the scope of this course).

Turbo-Convolutional Codes (Cont'd)

- Original Berrou's Feedback Decoder:



Turbo-Convolutional Codes (Cont'd)

- BER performance