Introduction to turbo processing

Kalle Ruttik

Communications Laboratory

Helsinki University of Technology

P.O.Box 2300, FIN-02015 HUT, FINLAND

Tel: +358 9 451235418; Fax: +358 9 4512359

`e-mail: kalle.ruttik@hut.fi`

February 4, 2005

Introduction

Channel coding theorem suggest that randomly generated code with appropriate distribution is likely good if the block length is high.

Problem: Decoding

- Without the structure codes with long block lenghts difficult to decode.

We can avoid the fully random codes since any code with the spectrum resembling random code is a good code.

The random like codes can be generated by interleaver.

Interleaver performs permutation of the bit sequence.

Permutation can be on the information bit sequence or on the parity bits.

Parallel concated convolutional codes (PCCC)

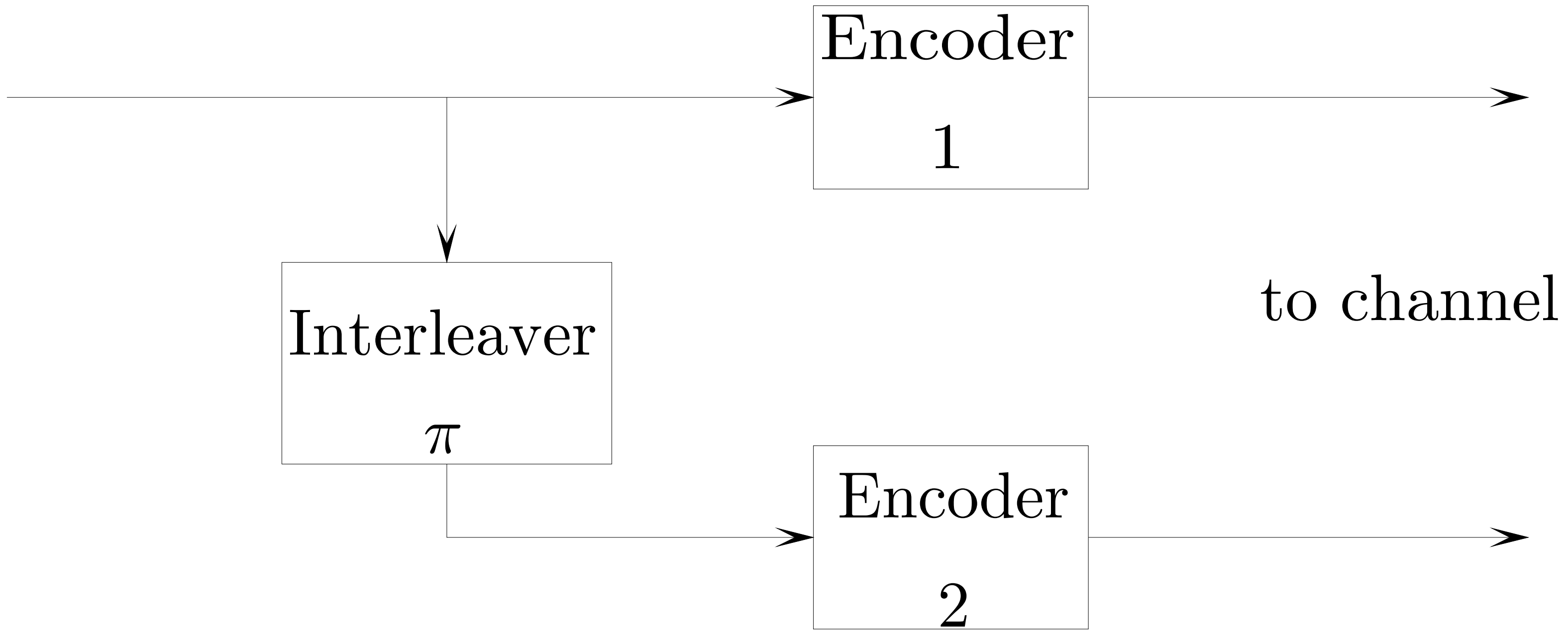


Figure 1: Encoder

Decoding turbo codes

- The parallel-concatenated convolutional code cannot be decoded by standard serial dynamic programming algorithm.
  - The number of states considered in trellis evaluation of two interleaved code is squared compared to the forward or backward calculations of one code.
  - Changing a symbol in one part of the turbo-coded codeword will affect possible paths in this part and also in the distant part where this bit is ”interleaved” in the other codeword.
  - Optimal path in one constituent codeword does not have to be optimal path in the other codeword.

Berrou approach

- Calculate the likelihood of each bit of the original dataword of being 0 or 1 accordingly to first code trellis.
- The second decoder uses the likelihood from the first decoder to calculate the new probability of the received bits but now accordingly to the received sequence of the second coder $y^{(2)}$.
- Bit estimate from the second decoder is feed again into first decoder.
- Instead of serially decoding each of the two trellises we decode both of them in parallel fashion.

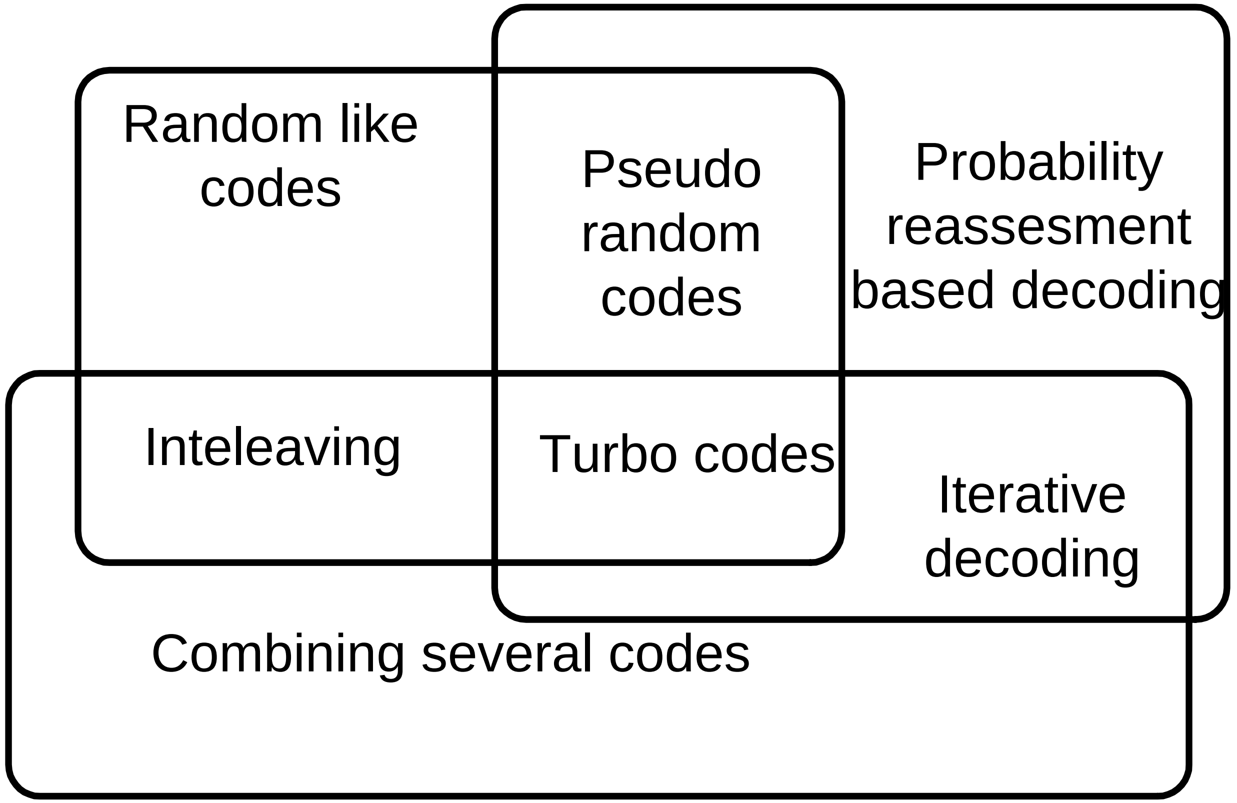


Figure 2: The ideas influencing evolution of turbo coding. Accordingly to fig. 1 from Battail97.

Code as a constraint

- The bit estimates calculated based on the coder structure are *aposteriori* probabilities of the bit constrained by the code.
- Contraint means that among of all possible bit sequences only some are allowed - they are possible codewords. The codewords limit the possible bit sequecnes.
- The aposteriori probability is calculated over the probabilities of possible codewords.
- The rule of the conditional probability

$$P\left(A|\,B\right) = \frac{P\left(A, B\right)}{P\left(B\right)}$$

where A correspond to a event that a certain bit in the codeword is 1 (or zero). B requires that the bit sequence is allowable codeword.

Example: repetition code

- We have three samples c_1,c_2,c_3.
- If to take the samples one by one they can be either zero or one.
- We have additional information: the samples are generated as a repetition code.
- Let denote the valid configurations as $S = \{(c_1, c_2, c_3)\}$.
- The set of possible codewords (the constraint set) is $S = \{(0,0,0), (1,1,1)\}$.
- In our case the *a posterior* probability for the sample $c_3 = 1$ is

$$p_3^{post} = \frac{\sum_{\substack{(c_1,c_2,c_3)\in S, \\ c_3=1}} P(c_1, c_2, c_3)}{\sum_{(c_1,c_2,c_3)\in S} P(c_1, c_2, c_3)}$$

In the numerator is summation over all configurations in S such that $c_3 = 1$, in the denominator is the normalisation, the sum of all the probabilities of all configurations in S.

- If the prior probabilities are independent the joint probability can be factorised

$$P\left(c_1, c_2, c_3\right) = P\left(c_1\right) P\left(c_2\right) P\left(c_3\right)$$

- The values for the prior probabilities could be acquired by measurements.
- A numerical excample: We have observed the samples and concluded that the samples have values 1 with the following probabilities
  $P(c_1 = 1) = \frac{1}{4}$,$P(c_2 = 1) = \frac{1}{3}$,$P(c_3 = 1) = \frac{1}{2}$,
  where c_i stands for the i-th observed sample and $i = 1, 2, 3$.
  What is the probability that the third bit is one?

- The probability of the third sample being one is

$$p_3^{post} = \frac{p_1 p_2 p_3}{p_1 p_2 p_3 + (1 - p_1)(1 - p_2)(1 - p_3)}$$

- Our equation in numerical values

$$p_3^{post} = \frac{\frac{1}{4} \cdot \frac{1}{3} \cdot \frac{1}{2}}{\frac{1}{4} \cdot \frac{1}{3} \cdot \frac{1}{2} + \left(1 - \frac{1}{4}\right)\left(1 - \frac{1}{3}\right)\left(1 - \frac{1}{2}\right)} = 0.1429$$

- The probability of $p_3(c_3 = 0)$

$$\begin{aligned} p_3(c_3 = 0) &= \frac{\left(1-\frac{1}{4}\right)\left(1-\frac{1}{3}\right)\left(1-\frac{1}{2}\right)}{\frac{1}{4}\cdot\frac{1}{3}\cdot\frac{1}{2}+\left(1-\frac{1}{4}\right)\left(1-\frac{1}{3}\right)\left(1-\frac{1}{2}\right)} \\ &= 0.8571 = 1 - p_3^{post} \end{aligned}$$

- By using likelihood ratio we can simplify further

$$\begin{aligned}\frac{p_3^{post}}{\left(1-p_3^{post}\right)} &= \frac{\sum_{\substack{(c_1,c_2,c_3)\in S,\\ c_3=1}} P(c_1,c_2,c_3)}{\sum_{\substack{(c_1,c_2,c_3)\in S\\ c_3=0}} P(c_1,c_2,c_3)} \\ &\Rightarrow \frac{p_1\cdot p_2\cdot p_3}{(1-p_1)\cdot(1-p_2)\cdot(1-p_3)} \\ &= \frac{p_1}{(1-p_1)}\cdot\frac{p_2}{(1-p_2)}\cdot\frac{p_3}{(1-p_3)}\end{aligned}$$

- In the logarithmic domain

$$\ln\frac{p\left(c_1=1\right)p\left(c_2=1\right)p\left(c_3=1\right)}{p\left(c_1=0\right)p\left(c_2=0\right)p\left(c_3=0\right)}$$

$$=\ln\frac{p\left(c_1=1\right)}{p\left(c_1=0\right)}+\ln\frac{p\left(c_2=1\right)}{p\left(c_2=0\right)}+\ln\frac{p\left(c_3=1\right)}{p\left(c_3=0\right)}$$

$$L^{post}(c_3)=L(c_1)+L(c_2)+L(c_3)$$

- Probability 0.5 indicates that nothing is known about the bit.

$$L(c_3) = 0$$

- Even if we do not know aything about the bit but we know probabilties for the other bits we can calculate the aposteriori probability for the unknown bit.

$$L^{post}(c_3) = L(c_1) + L(c_2)$$

- The posteriori probability calculation for a bit can be separated into two parts:
  - part describing prior probability
  - part impacted by the constraint imposed by the code. This later part is calculated only based on the probabilities of other bits.

Parity-check

- Assume now that there can be even number of ones among the three samples $S = \{(0,0,0), (1,1,0), (1,0,1), (0,1,1)\}$
- The first two bits are either 0 or 1 the third bit is calculated as XOR of first two bits.
- Assume the measured probability for the first bit is 0.5
- The posterior probability of the first sample is

$$p_1^{post} = \frac{p_2 (1 - p_3) + (1 - p_2) p_3}{p_2 \cdot p_3 + p_2 \cdot (1 - p_3) + (1 - p_2) \cdot p_3 + (1 - p_2) \cdot (1 - p_3)}$$

- The probability that the first sample is 1 is given by the probability that exactly one of the other two samples is 1.
- The probability that the first sample is 0 is given by the probability that both other samples are 0 or both of them are 1.

- Likelihood ratio for the third sample posterior probability is

$$\frac{p^{post}\left(c_1 = 1\right)}{p^{post}\left(c_1 = 0\right)} = \frac{p_1\left(1 - p_2\right) + \left(1 - p_1\right)p_2}{p_1 p_2 + \left(1 - p_1\right)\left(1 - p_2\right)}$$

$$= \frac{\frac{1}{4}\left(1 - \frac{1}{3}\right) + \left(1 - \frac{1}{4}\right)\frac{1}{3}}{\frac{1}{4}\frac{1}{3} + \left(1 - \frac{1}{4}\right)\left(1 - \frac{1}{3}\right)} = 0.595$$

Parity check in log domain

- In logarithmic domain can be separated the prior for the bit and information from the other bits.

$$\ln\left(\frac{p_1(c_1=1)\cdot p\left((c_2\oplus c_3)=1\right)}{p_1(c_1=0)\cdot p\left((c_2\oplus c_3)=0\right)}\right)=$$

$$\ln\left(\frac{p_1(c_1=1)}{p_1(c_1=0)}\cdot\frac{p_2(c_2=1)p_3(c_3=0)+p_2(c_2=0)p_3(c_3=1)}{p_2(c_2=0)p_3(c_3=0)+p_2(c_2=1)p_3(c_3=1)}\right)=$$

$$L_1(c_1)+\ln\left(\frac{p_2(c_2=1)p_3(c_3=0)+p_2(c_2=0)p_3(c_3=1)}{p_2(c_2=0)p_3(c_3=0)+p_2(c_2=1)p_3(c_3=1)}\right)$$

Probabilities in log domain

- Here we give the probability calculation folmulas for the binary code, GF(2).
- The log-likelihood ratio (LLR) of c is

$$\begin{aligned} L(c) &= \ln \frac{p(c=1)}{p(c=0)} = \ln \frac{p(c=1)}{1-p(c=1)} \\ p(c=1) &= \frac{e^{L(c)}}{1+e^{L(c)}} = \frac{1}{1+e^{-L(c)}} \Rightarrow \\ p(c=0) &= 1-p(c=1) = \frac{1}{1+e^{L(c)}} = \frac{e^{-L(c)}}{1+e^{-L(c)}} \end{aligned}$$

Incorporating proababilities from different encoders

- Often we have two or more independent ways to calculate the aposteriori probability of the bit.
- The bit estimates from different sources are similar to repetition code. All the estiamtes have to have the same bit value.
- Because all the estiamtes have to be either 0 or 1 in log domain we can simple sum together the loglikelihood ratios from different estimations.

Single Parity Check Product Code (Example)

- SPC product code - a simple example of a concatenated code
- Two separate coding steps - horizontal, vertical

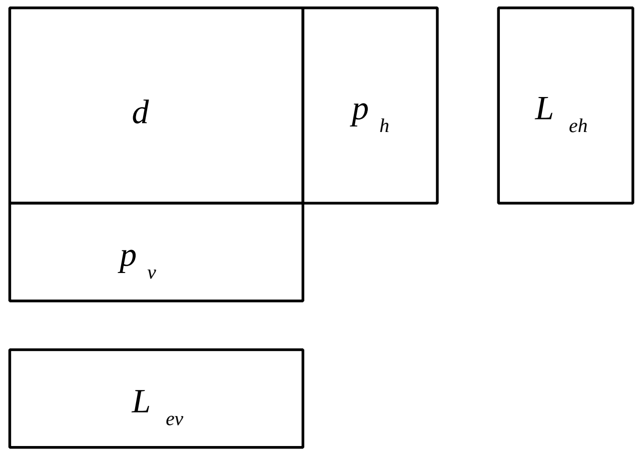


Figure 3: Two dimensional product code

$k_1 \times k_2$ data array d; $n_2 - k_2$ parity bits p_h; $n_1 - k_1$ parity bits p_v, L_{eh}, L_{ev} stand for the extrinsic LLR values learned from the horizontal and vertical decoding steps.

Numerical example

d_1	d_2	d_3	d_4	p_{1h}	p_{2h}	p_{1v}	p_{2v}
$+1$	$+1$	$+1$	-1	$+1$	-1	$+1$	-1
0.25	2.0	5.0	1.0	1.0	-1.5	2.0	-2.5

d_1	d_2	p_{1h}
d_3	d_4	p_{2h}
p_{1v}	p_{2v}	

$\Rightarrow$

$+$	$+$	$+$
$+$	$-$	$-$
$+$	$-$	

$\Rightarrow$

$L_c(x_1) = 0.25$	$L_c(x_2) = 2.0$	$L_c(x_{12}) = 1.0$
$L_c(x_3) = 5.0$	$L_c(x_4) = 1.0$	$L_c(x_{34}) = -1.5$
$L_c(x_{13}) = 2.0$	$L_c(x_{24}) = -2.5$	

Decoding Algorithm

1. Set prior information $p(\hat{d}) = 0.5$ for all d $L(\hat{d}) = 0$.
2. Sum the observed probability and prior probability $L_c(d_{\#}) + L(d_{\#})$.
3. Decode horizontally. Obtain the bit probabilities based on the constraint posed by the horisontal code. The result is called horizontal extrinsic information.
   - The parity bit is generated by the xor of information bits in the horisontal line.
   - The extrinsic information can be generated as the *aposteriori* probability calculation accordingly to parity check. For example

$$L_h^{extr}(d_1) = \ln\left(\frac{p(d_2 = 1)p(p_{1h} = -1) + p(d_2 = -1)p(p_{1h} = 1)}{p(d_2 = -1)p(p_{1h} = -1) + p(d_2 = 1)p(p_{1h} = 1)}\right)$$

4. Set $L(\hat{d}_1) = L_h^{extr}(d_1)$.
5. Combine the *aposteriori* horisontal and priori information for each bit. For example for the bit 1

$$L^{combined}(d_1) = L_c(d_1) + L(\hat{d}_1)$$

$$p^c(d_1 = 1) = \frac{e^{L^{combined}(d_1)}}{\left(1 + e^{L^{combined}(d_1)}\right)}$$

6. Decode vertically. In computations instead of p use p^c. Obtain the vertical extrinsic information for each bit. For example

$$L_v^{extr}(d_1) = \ln\left(\frac{p^c(d_3 = 1)p(p_{1v} = -1) + p^c(d_3 = -1)p(p_{1v} = 1)}{p^c(d_3 = -1)p(p_{1v} = -1) + p^c(d_3 = 1)p(p_{1v} = 1)}\right)$$

7. If the interations have not finished
   - combine the information of the *aposteriori* vertical $L_h^{extr}(d_\#)$ and priori information $L(d_\#)$ for each bit.
   - go back to stage 2.

   else
   - Combine all the information for the bit the priori *aposteriori* vertical and horisontal.
     $L^d(d_\#) = L_c(d_\#) + L_v^{extr}(d_\#) + L_h^{extr}(d_\#)$
   - Compare the likelihood ratio of the bit with the decision level (0).

The soft output for the received signal corresponding to data d_i

$$L(\hat{d}_i) = L_c(x_i) + L(\hat{d}_i) + L_h^{extr}(d)$$

Decode horizontally

$$\begin{aligned} L_h^{extr}(d_1) &= \ln\left(\frac{p(d_2{=}1)p(p_{1h}{=}{-1}){+}p(d_2{=}{-1})p(p_{1h}{=}1)}{p(d_2{=}{-1})p(p_{1h}{=}{-1}){+}p(d_2{=}1)p(p_{1h}{=}1)}\right) \\ &= 0.74 = newL(\hat{d}_1) \end{aligned}$$

$$L_h^{extr}(d_2) = +0.12 = newL(\hat{d}_2)$$

$$L_h^{extr}(d_3) = -0.60 = newL(\hat{d}_3)$$

$$L_h^{extr}(d_4) = -1.47 = newL(\hat{d}_4)$$

$$\begin{array}{|c|c|}\hline +0.25 & +2.0 \\ \hline +5.0 & +1.0 \\ \hline \end{array} + \begin{array}{|c|c|}\hline +0.74 & +0.12 \\ \hline -0.60 & -1.47 \\ \hline \end{array}$$ after 1st horizontal decoding

Decode vertically

$$L_h^{extr}(d_2) = +0.33 = newL(\hat{d}_2)$$

$$L_h^{extr}(d_2) = +0.09 = newL(\hat{d}_2)$$

$$L_h^{extr}(d_3) = -0.36 = newL(\hat{d}_3)$$

$$L_h^{extr}(d_4) = -0.26 = newL(\hat{d}_4)$$

$$\begin{array}{|c|c|}\hline +0.33 & +0.09 \\ \hline +0.36 & -0.26 \\ \hline \end{array}$$ Extrinsic information after 1st vertical decoding

Soft output after 1st iteration $L(\hat{d}) = L_c(x) + L_{eh}(d) + L_{ev}(d)$

$$\begin{array}{|c|c|}\hline +0.25 & +2 \\ \hline +5.0 & +1 \\ \hline \end{array} + \begin{array}{|c|c|}\hline +0.74 & +0.12 \\ \hline -0.60 & -1.47 \\ \hline \end{array} + \begin{array}{|c|c|}\hline +0.33 & +0.09 \\ \hline +0.36 & -0.26 \\ \hline \end{array} = \begin{array}{|c|c|}\hline +1.31 & +2.20 \\ \hline +4.75 & -0.74 \\ \hline \end{array}$$

We can see an iterative process:

1 Decode first code and calculate extrinsic information for each bit.
   - In first iteration the information from other code is zero.

2 Decode the second code by using extinsic information from the first decoder.

3 Return to the first step by using the extrinsic information from the second decoder.

Turbo Decoding

- Turbo codes are constructed by applying two or more component codes to different interleaved versions of the same information sequence.
- The idea of turbo decoding is to pass information from output of one decoder to the input of the other decoder and to iterate this process several times to produce better decision.
- Decoding algorithm should exchange soft decisions rather than hard decisions.
- The optimal soft output should be the a posteriori probability (APP), the probability of the received bits or sequence conditioned on the received signal.

Turbo Principle

Iterative exchange of soft information between different blocks in a receiver nowadays is successfully applied (beyond channel coding) for a wide range of communications problems:

- Combined equalization/estimation and error correction decoding
- Iteratively improved synchronization
- Combined multiuser detection and error correction decoding
- (Spatial) diversity combining for coded systems in the presence of multiple-access interference (MAI) or inter-symbol interference (ISI)

$\Rightarrow$ Iterative Receiver Concept

Turbo Equalization

ISI channel is the inner code in the serial concatenation.
ISI channel may be seen as a rate 1 code defined over the field of real numbers

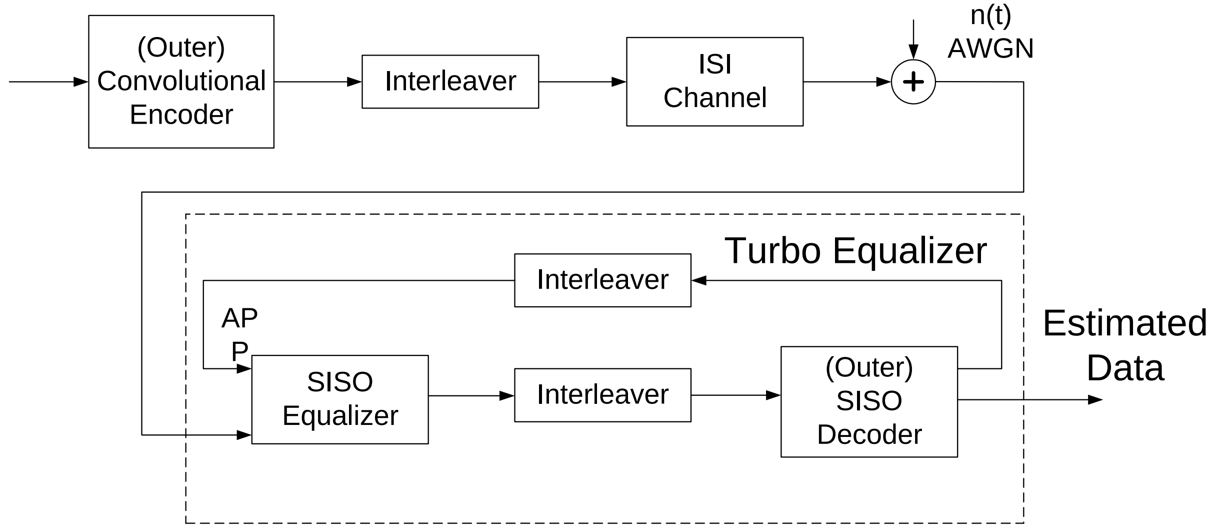


Figure 4: Turbo equalisation.

Turbo Multiuser Detection (1)

Multiple-access interference (MAI) channel is the inner code of a serial concatenation

MAI channel may be seen 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 users in the system.

Turbo Multiuser Detection (2)

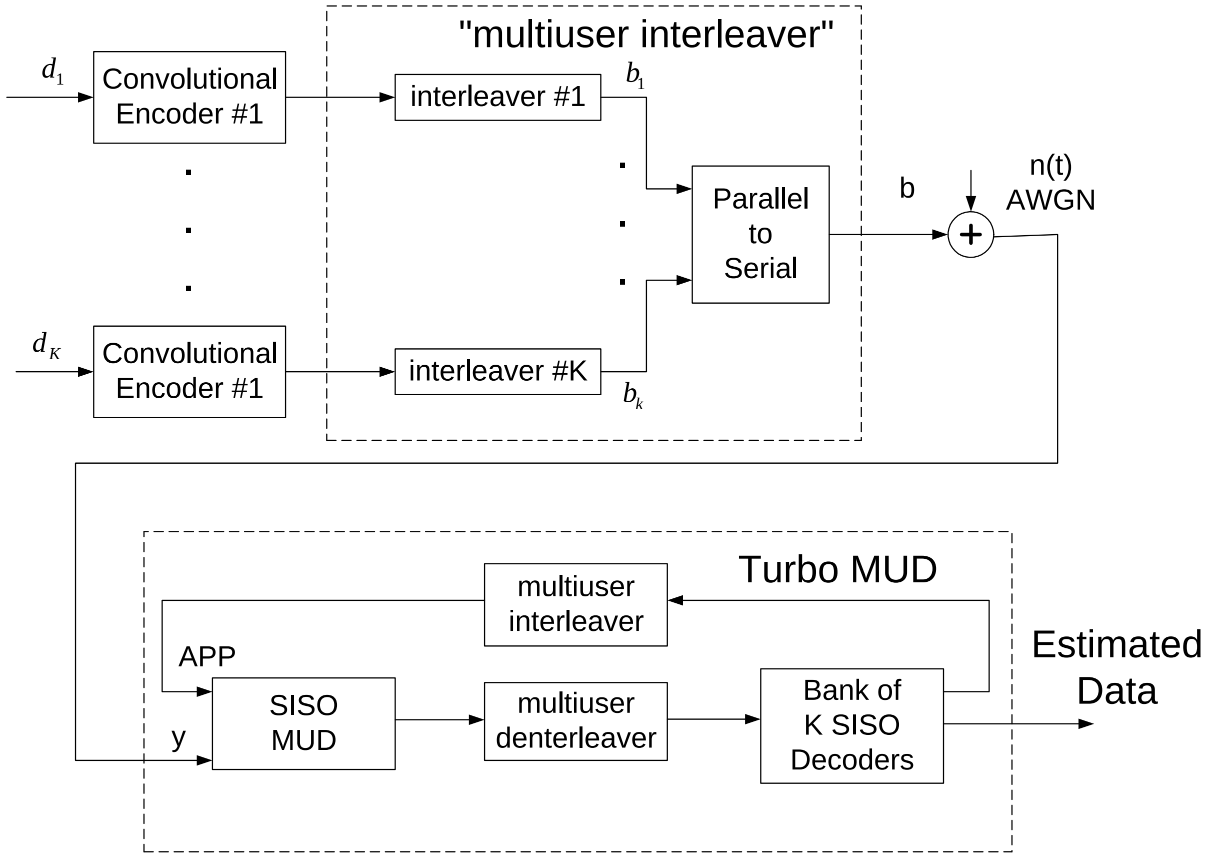


Figure 5: Turbo multi user detection.

Soft Channel Outputs

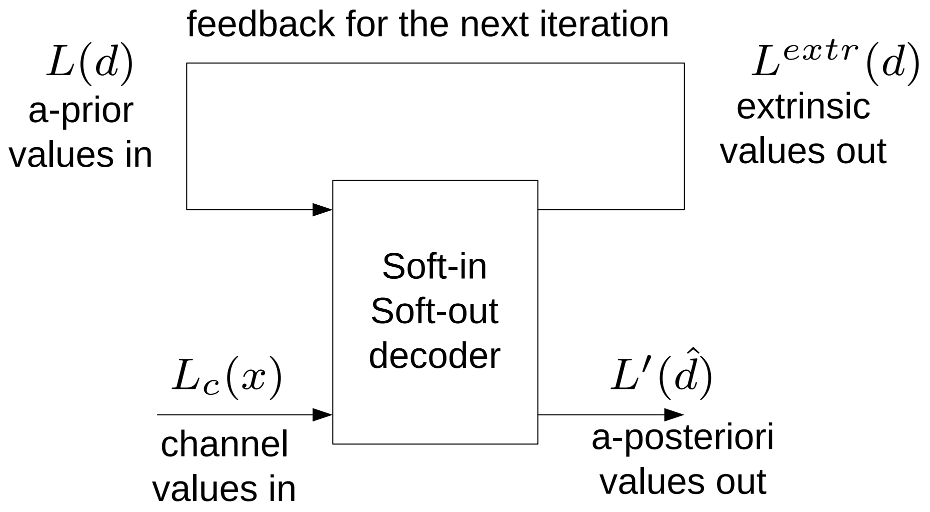
- Let binary elements 0 and 1 present data $d = 1$ and $d = -1$.
- LLR of d conditioned on $x = d + n$ (data + noise)
  $L(d|x) = \ln \dfrac{p(d = +1|x)}{p(d = -1|x)}$
  $p(d = 1|x)$ is the conditional a posteriori probability (APP).
- Using Bayes theorem:
  $L(d|x) = \ln \dfrac{p(x|d = +1)}{p(x|d = -1)} + \ln \dfrac{p(d = +1)}{p(d = -1)} = L(x|d) + L(d)$
  where $L(x|d)$ is the LLR of channel measuremetns of x under the alternate conditions $d = \pm 1$.
- Assuming AWGN with $n \in N(0, \sigma^2)$ the probability of the matched filter output is

$$p(x|d = +1) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{E_b}{2\sigma^2}(x-a)^2\right)}$$

$$\begin{aligned}
L(d_k|x_k) &= \ln\left(\frac{\exp\left(-\frac{E_b}{2\sigma^2}(x_k-a)^2\right)}{\exp\left(-\frac{E_b}{2\sigma^2}(x_k+a)^2\right)}\right) + \ln\frac{p(d_k=+1)}{p(d_k=-1)} \\
&= \left(\left(-\frac{E_b}{2\sigma^2}(x_k-a)^2\right) - \left(-\frac{E_b}{2\sigma^2}(x_k+a)^2\right)\right) + \ln\frac{p(d_k=+1)}{p(d_k=-1)} \\
&= \tfrac{E_b}{2\sigma^2}\cdot 4a\cdot x_k + \ln\tfrac{p(d_k=+1)}{p(d_k=-1)} \\
&= L_c(x_k) + L(d_k)
\end{aligned}$$

where $L_c(x_k) = \frac{2E_b}{\sigma^2}\cdot 4a\cdot x_k$,
for binary symmetric channel (BSC) $L_c(x_k)$ is channel LLR.

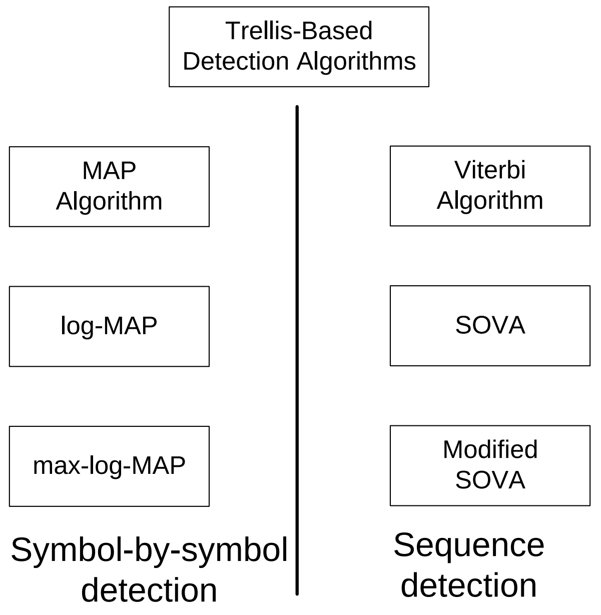
SISO Decoder



Soft decision $L(\hat{d})$ is a real number: the sign of it gives the hard decision and the magnitude denotes the reliability of that decision.

Decoder improves decision reliability: $L'(\hat{d}) = L_c(x) + L(d) + L_e(d)$

Algorithms for Iterative (Turbo) Data Processing



Requirements

Accept soft-inputs in the form of a priori probabilities or log-likelihood ratios

Produce APP for output data

Soft-Input Soft-Output

- MAP: Maximum A Posteriori (symbol-by-symbol)
- SOVA: Soft Output Viterbi Algorithm

Approximations

- The loglikelihood after equal operation

$$L\left((d_1 = d_2)\right) = \ln\left(e^{(L(d_1)+L(d_2))}\right) = (L(d_1) + L(d_2))$$

- The loglikelihood after modulo 2 operation

$$L(d_1 \oplus d_2) = \frac{1 + \exp(L(d_1))\exp(L(d_2))}{\exp(L(d_1)) + \exp(L(d_2))}$$

- We can simplify the equation by selecting only the minimum

$$L(d_1 \oplus d_2) \approx sign(L(d_1)L(d_2))\min\{|L(d_1)|, |L(d_2)|\}$$

- The simplified version allows to calculate the output probability by simple selection.
  Define addition for LLR: $L(d_1) \boxplus L(d_2) \triangleq L(d_1 \oplus d_2)$
  with rules $L(d) \boxplus 0 = 0$, $L(d) \boxplus -\infty = -L(d)$
- By induction
  $\sum_i \boxplus L(d_i) = L(\sum_i \oplus d_i) \approx \prod_i sign(L(d_i))\min_i\{|L(d_i)|\}$