Concatenated Coding

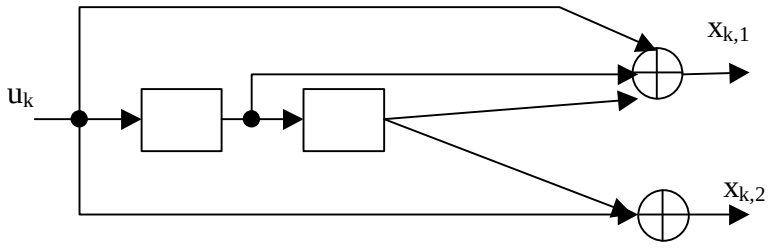
- **Parallel Concatenated Convolutional Codes (PCCC) or "Turbo" Codes**

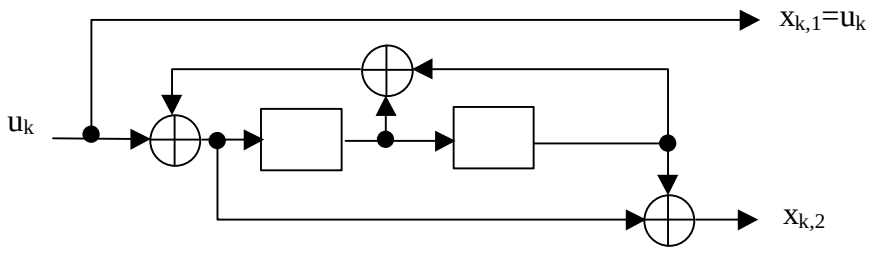
First Paper: "NEAR SHANNON LIMIT ERROR-CORRECTING CODING AND DECODING: TURBO – CODES(1)" by
Claude Berrou, Alain Glavieux and **Punya Thitimajshima**
Ecole Nationale Superieure des Telecommunications (ENST) de Bretagne , France

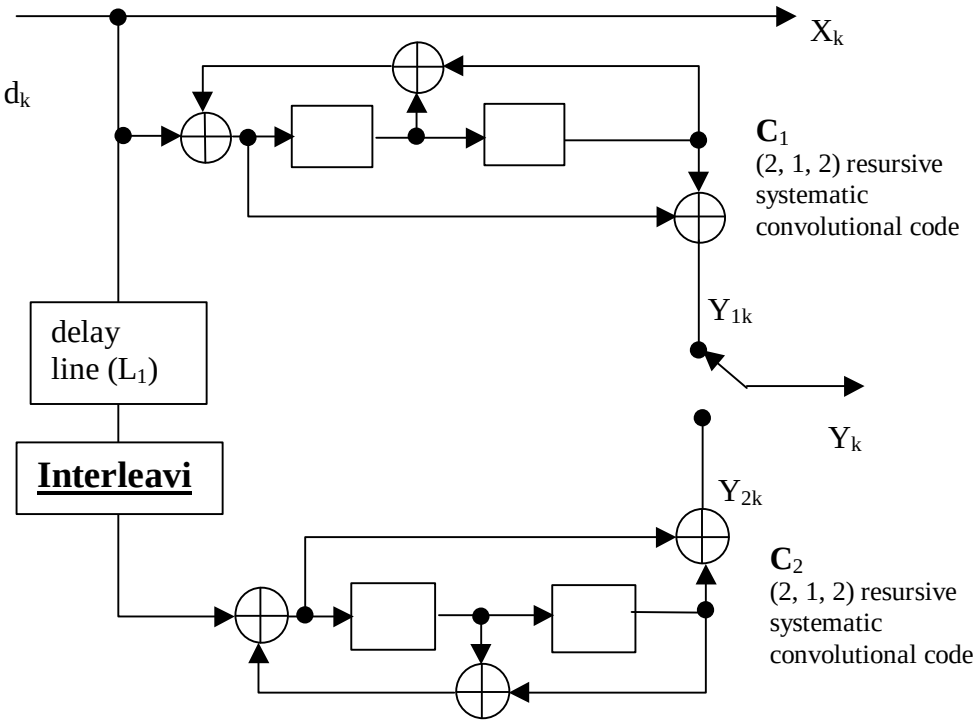
International Communications Conference (**ICC**), Geneva, Switzerland May **1993**, pp. 1064-1070.

- use systematic codes – nonsystematic ones have equivalent forms
- systematic codes with feed forward encoders produce less powerful codes. Convolutional codes in their systematic feedback form are equivalent to the nonsystematic form in distance and nearest neighbor path properties. However at low signal to noise ratio (SNR), the bit error rate (BER) is slightly better.

Figures of FB-CC and FF-CC for R=1/2 v=2.







We can now decode the information sequence or move to the second decoder which will decode for the same information sequence but in the interleaved form. Now we'll make use of the $L_e(i)$ generated by the first encoder.

This can be given as *a priori* ($L_e(i)$ as $L_{I2}(i)$ **-after interleaving**) information to the second decoder.

So effectively we can **interleave** $L_{sys}(i)$ and $L_e(i)$ and feed it to the second decoder. $p_{ap}(i_t=-1)$ and $p_{ap}(i_t=1)$ can be used associated with γ, in the calculation of α,β for the second decoder .

Thus the output of the second decoder will be of the form

$$L_{app}(i) = L_{sys2}(i) + L_{I2}(i) + L_{e2}(i)$$

This finishes the first or **initial full decoding step- step 0**.

$L_{e2}(i)$ reflects the extrinsic information generated by the second decoder. Now in a similar manner as earlier we can give - feedback the first decoder with $L_{e2}(i)$ (γ to calculate α,β) after **deinterleaving** as new *a priori* $L_{I1}(i)$ information to the first decoder.

<u>Decoding step 1</u>

First Decoder

$L_{app}(i) = L_{sys}(i) + L_{I1}(i) + L_{e1}(i)$

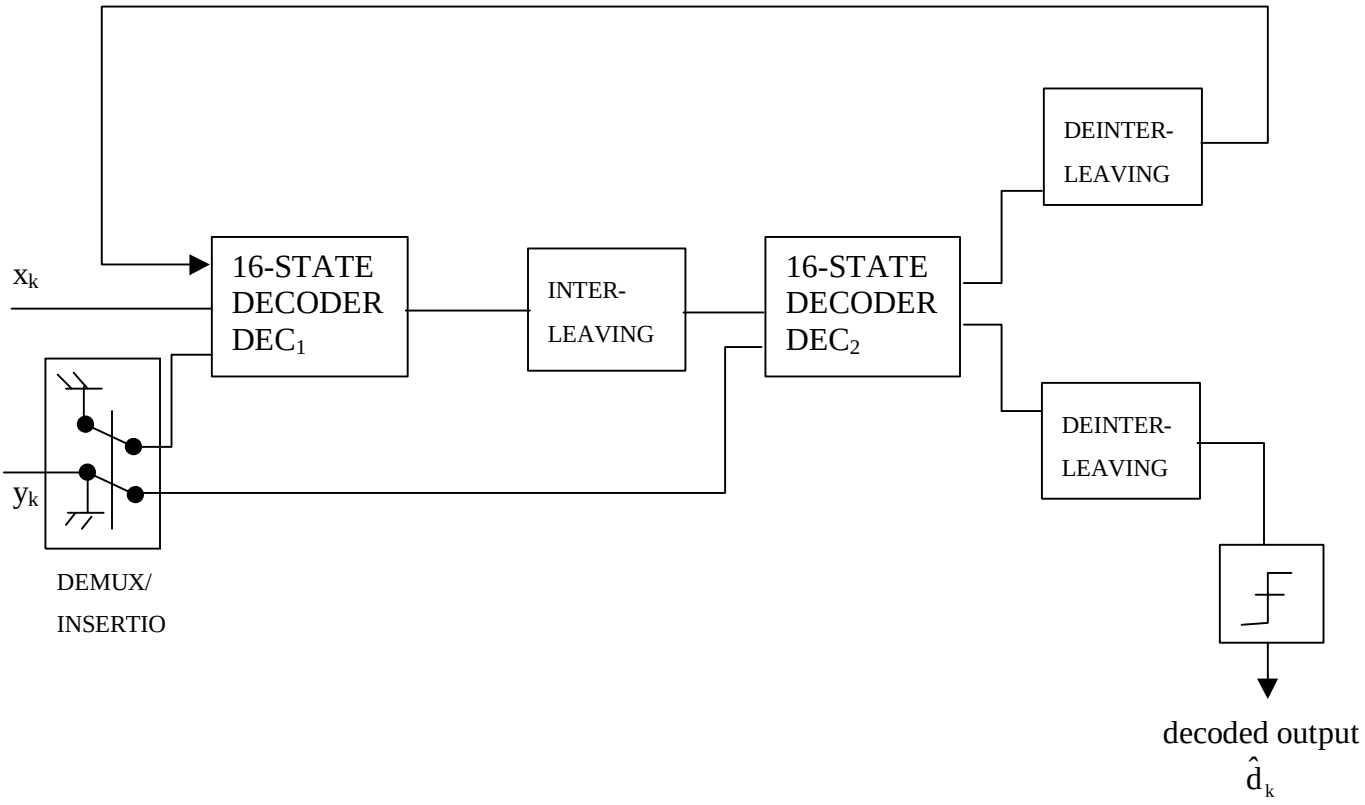
Second Decoder (after interleaving $L_{e1}(i)$)

$L_{app}(i) = L_{sys}(i) + L_{I2}(i) + L_{e2}(i)$

Then from decoder two feedback $L_{e2}(i)$ as before to the first decoder after **deinterleaving – and the process repeats**.

This is known as **iterative** decoding or **turbo** decoding.

- Remember it is always the **extrinsic** information that we pass to the next decoder as new a priori. We don't feedback *a priori* or *intrinsic* information- why? – that was already generated by the earlier decoder so no point in feeding back the same information.



The $p_I(i_t=\pm 1)$ needed to calculate γ can also be obtained from $L_I(i)$ which will in turn be obtained from $L_e(i)$ of the earlier decoding stage.

Reference: "**Iterative Decoding of Binary Block and Convolutional Cod**es" by Joachim Hagenauer, Elke Offer, and Lutz Papke, IEEE Tans. Info. Theory, Vol. 42, No.2, March 1996, pp. 429-445.

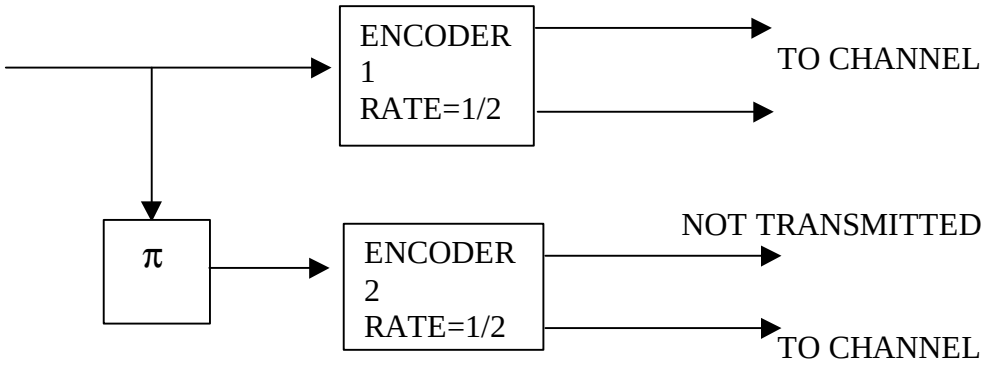
Text Book Chapters:

Bossert's: Chapter 8 on Convolutional Codes and Appendix A on SCC, PCC
Fundamentals of Convolutional Coding: Chapter 7- Iterative Decoding

PCCC –

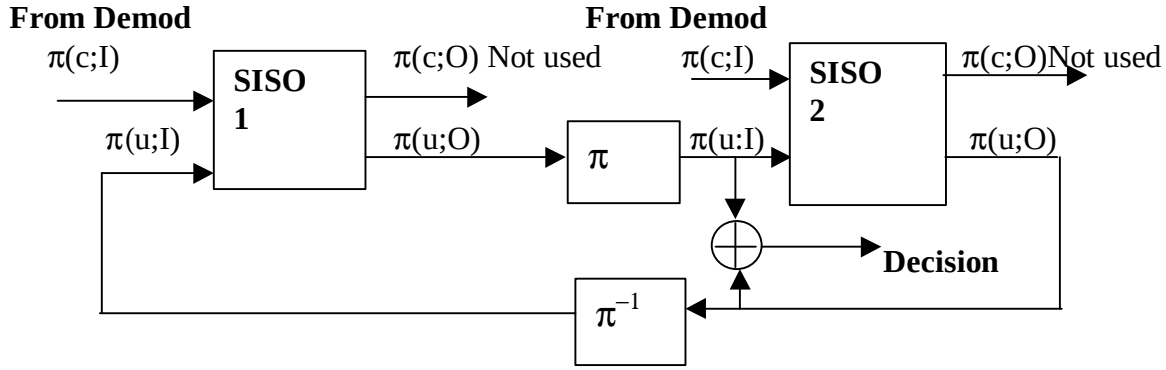
Reference: A soft-Input Soft-Output maximum A Posteriori (MAP) Module to Decode Parallel and Serial Concatenated Codes
S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara
TDA Progress Report 42-127, November 1996 from JPL web site

Block diagram: **Encoder**



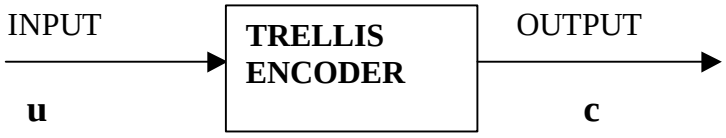
Decoder

π(.) represents log probabilities or working in the log domain.



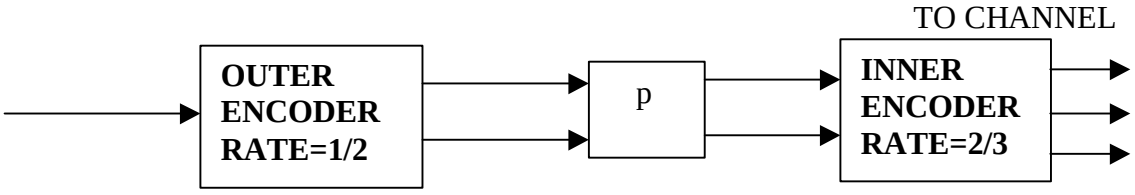
SISO – soft-input, soft-output Module: implements MAP algorithm

The general Trellis Encoder:

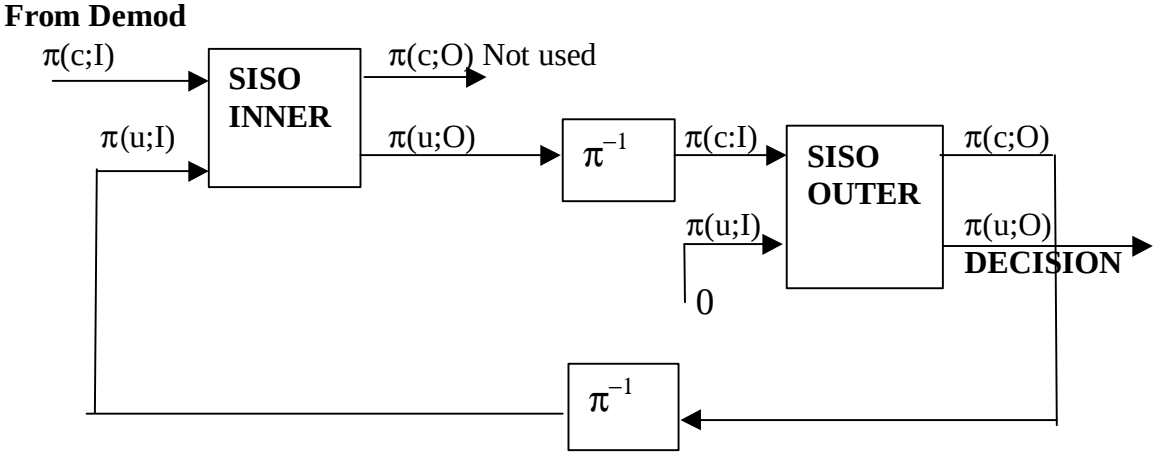


SCCC – Serial Concatenated Convolutional Codes

Encoder:



Decoder:

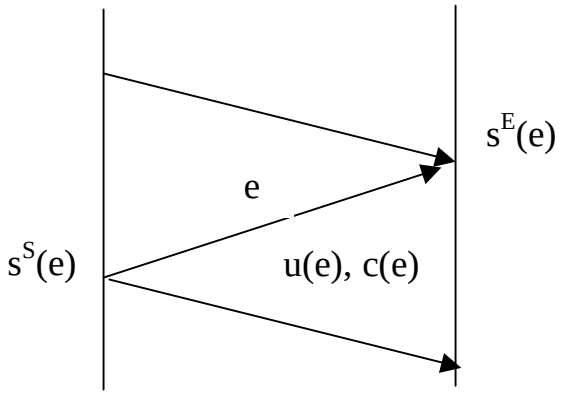


Differences between PCCC and SCCC:

PCCC: Updated probabilities (extrinsic) of **code** symbols are **never** used by the decoding algorithm

SCCC: Both, updated probabilities (extrinsic) of the input and code symbols are used in the decoding algorithm

Decoding algorithm is – Additive SISO algorithm (A-SISO) working in the log domain



An Edge (e) of the Trellis Section

The following functions are associated with each edge e

- The starting state $s^S(e)$
- The ending state $s^E(e)$
- The input symbol
- The output symbol

The relationship between these functions depends on the particular encoder. As an example, in the case of systematic encoders, (s^E(e),c(e)) also identifies the edge since u(e) is uniquely determined by c(e).

Here it is only assumed that the pair (s^S(e), u(e)) uniquely identifies the ending state s^E(e) – this assumption is always verified , as it is equivalent to say that – given the initial trellis state, there is a one-to-one correspondence between input sequences and state sequences.

The Additive SISO Algorithm (A-SISO)

$$a_k(s) = \log\left[\sum_{e:s^E(e)=s} \exp\{a_{k-1}[s^S(e)] + p_k[u(e);I] + p_k[c(e);I]\}\right], k = 1,2,...n$$

$$b_k(s) = \log\left[\sum_{e:s^S(e)=s} \exp\{b_{k+1}[s^E(e)] + p_{k+1}[u(e);I] + p_{k+1}[c(e);I]\}\right],$$
$$k = n-1,..,0$$

These are forward and backward recursions. At time k, **the output (extrinsic) probability distributions** are computed as (approximately)

$$p_k(c;O) = \log\left[\sum_{e:c(e)=c} \exp\{a_{k-1}[s^S(e)] + p_k[u(e);I] + b_k[s^E(e)]\}\right]$$

$$p_k(u;O) = \log\left[\sum_{e:u(e)=u} \exp\{a_{k-1}[s^S(e)] + p_k[c(e);I] + b_k[s^E(e)]\}\right]$$

with initial values $\alpha_0(s) = 0$ for $s=S_0$ otherwise $\alpha_0(s) = -\infty$

similarly $\beta n(s) = 0$ for $s=S_n$ otherwise $\beta_n(s) = -\infty$

So we replace **log** and **exp** with **maximum** values. Thus we have the following set of equations.

$$\alpha_k(s) = \max_{(e:s^E(e)=s)} \{\alpha_{k-1}[s^S(e) + \pi_k[u(e);I] + \pi_k[c(e);I]\},\ k=1,2..,n$$

$$\beta_k(s) = \max_{(e:s^S(e)=s)} \{\beta_{k+1}[s^E(e) + \pi_{k+1}[u(e);I] + \pi_{k+1}[c(e);I]\},\ k=n-1,\ldots,0$$

$$\pi_k(c;O) = \max_{(e:c(e)=c))} \{\alpha_{k-1}[s^S(e) + \pi_k[u(e);I] + \beta_k[s^E(e)]\}$$

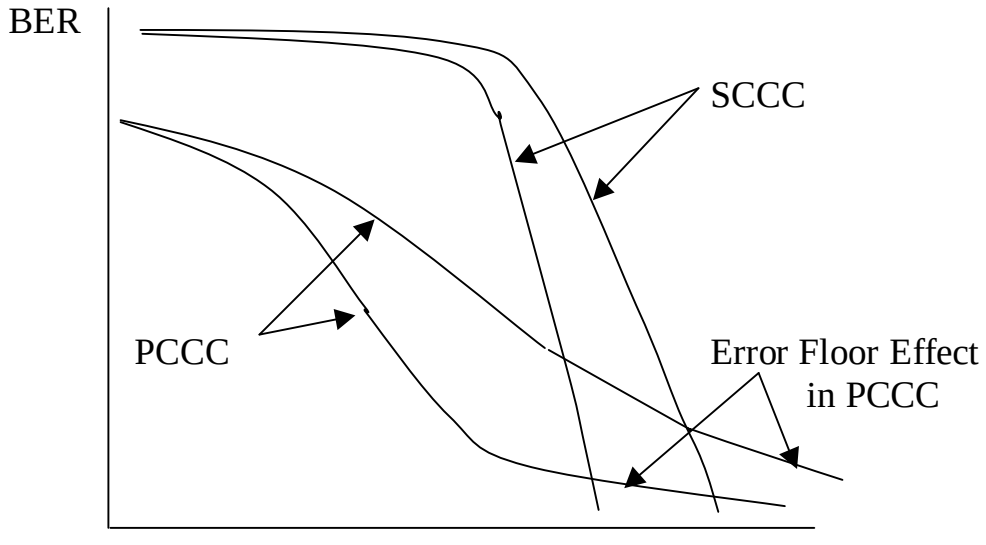
$$\pi_k(u;O) = \max_{(e:u(e)=u))} \{\alpha_{k-1}[s^S(e) + \pi_k[c(e);I] + \beta_k[s^E(e)]\}$$

Generally for serial concatenated codes:

OUTER CODE- NON RECURSIVE or NON-Feed back , Non systematic

INNER CODE- RECURSIVE SYSTEMATIC or Feed-back Systematic

It is seen that the performance of SCCC is usually better than that of PCCC.



The summations involved in the algorithm are calculated using trellis edges, rather than using pairs of states. This makes the algorithm general and capable of dealing with parallel edges – **suitable for TCM**.

The A-SISO algorithm at Bit Level

Consider a rate ½ convolutional encoder. U_k input and $C_{1,k}$ and $C_{2,k}$ output bits at time k, taking values {0,1}. Therefore on the trellis edges at time k we have $u_k(e)$, $c_{1,k}(e)$, $c_{2,k}(e)$. Drop k for simplicity.

Define the reliability (LLR) of a bit Z taking values {0,1} at time k as

$$l_k[Z;.] \equiv \log \frac{P_k[Z=1;.]}{Pk[Z=0;.]} = p_k[Z=1;.] - p_k[Z=0;.]$$

$$\alpha_k(s) = \max_{(e:s^E(e)=s)} \{\alpha_{k-1}[s^S(e) + u(e)\lambda_k[u;I] + c_1(e)\lambda_k[C_1;I] + c_2(e)\lambda_k[C_2;I]\} + h_{\alpha k}$$

$$\beta_k(s) = \max_{(e:s^S(e)=s)} \{\beta_{k+1}[s^E(e) + u(e)\lambda_{k+1}[u;I] + c_1(e)\lambda_{k+1}[C_1;I] + c_2(e)\lambda_{k+1}[C_2;I]\} + h_{\beta k}$$

with initial values $\alpha_0(s) = 0$ if $s=S_0$ and $\alpha_0(s) = -\infty$ otherwise and $\beta_n(s)=0$ if $s=S_n$ and $\beta_n(s) = -\infty$ otherwise. $h_{\alpha k}$ and $h_{\beta k}$ are normalization constants.

For the inner decoder, which is connected to the AWGN channel, we have

$\lambda_k[C_1;I]= (2A/\sigma^2)r_{1,k}$ $\lambda_k[C_2;I]= (2A/\sigma^2)r_{2,k}$ where

$r_{i,k} = A(2Z_i - 1) + i_{,k}$, i=1,2

is the received samples at the output of the matched filter, $c_i \in \{-1,1\}$ and $n_{i,k}$ is the zero mean independent identically distributed (i.i.d.) Gaussian noise samples with variance σ^2.

The **extrinsic bit information** for U, C_1 and C_2 can be obtained as

$$\lambda_k(U;O) = \max_{\mathbf{(e:u(e)=1)}} \{\alpha_{k-1}[s^S(e) + c_1(e)\lambda_k[C_1;I] + c_2(e)\lambda_k[C_2;I] + \beta_k[s^E(e)]\} - \max_{\mathbf{(e:u(e)=0)}} \{\alpha_{k-1}[s^S(e) + c_1(e)\lambda_k[C_1;I] + c_2(e)\lambda_k[C_2;I] + \beta_k[s^E(e)]\}$$

$$\lambda_k(C_1;O) = \max_{\mathbf{(e:c1(e)=1)}} \{\alpha_{k-1}[s^S(e) + u(e)\lambda_k[U;I] + c_2(e)\lambda_k[C_2;I] + \beta_k[s^E(e)]\} - \max_{\mathbf{(e:c1(e)=0)}} \{\alpha_{k-1}[s^S(e) + u(e)\lambda_k[U;I] + c_2(e)\lambda_k[C_2;I] + \beta_k[s^E(e)]\}$$

$$\lambda_k(C_2;O) = \max_{\mathbf{(e:c2(e)=1)}} \{\alpha_{k-1}[s^S(e) + u(e)\lambda_k[U;I] + c_1(e)\lambda_k[C_1;I] + \beta_k[s^E(e)]\} - \max_{\mathbf{(e:c2(e)=0)}} \{\alpha_{k-1}[s^S(e) + u(e)\lambda_k[U;I] + c_1(e)\lambda_k[C_1;I] + \beta_k[s^E(e)]\}$$

Parameters for PCCC

In general for a Convolutional code (CC) w_{min} is defined as the minimum information weight in the error events of the CC.

$w_{min} = 1$ for non recursive codes $w_{min} = 2$ for recursive codes.

The error coefficient (in p_b(error)) interleaving gain for a PCCC with large interleaving length goes as $N^{1-w\,min}$, where N is the interleaver length.

Thus for recursive CC's the interleaving gain (or BER reduction) goes as 1/N. On the other hand all nonrecursive CC and block codes have $w_{min} = 1$, so such codes are not useful in Parallel Concatenated Codes.

The next most important constituent code parameter is z_{min}, the minimum parity-check weight in the code sequences with w=2. For a large range of SNR, the behaviour of PCCC is determined by the "**effective free distance**"

$d_{free,eff} = 2 + 2\, z_{min}$.

It is possible to achieve $z_{min} = (n-1)(2^{v-1} + 2)$ with a rate 1/n recursive CC with memory v.

PCCC's exhibit error floor effect (defined as the change of slope of BER curve for Turbo codes) at lower BER. Initially the coding gain increases with the number of iterations. However, after a number of iterations (10-12 in AWGN) the performance improvement is marginal.

Extensions to consider:

1. Turbo codes with multilevel modulation
2. Turbo TCM
3. Iterative Equalization and Decoding – Turbo Equalization.