Concatenated Codes with Interleaving or Turbo Codes.

Laura Cottatellucci

EURECOM

laura.cottatellucci@eurecom.fr

Outlines

I. More on Convolutional Codes: Systematic Feedback Encoders

II. Historical Notes: Concatenated Codes

III. Concatenated Codes with Interleaving: Four Basic Schemes

IV. Spectral Thinning

V. Notes on Soft Decoding

VI. Iterative Decoding

VII. Performance Analysis at High SNR

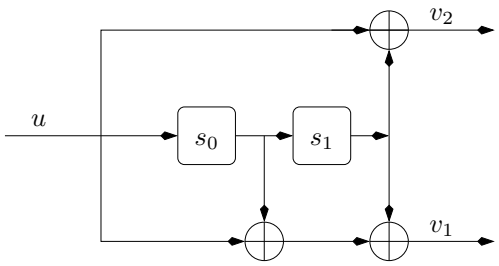
VIII. Performance Analysis at Low SNR: Exit Charts

IX. References

Systematic Feedback Convolutional Encoders

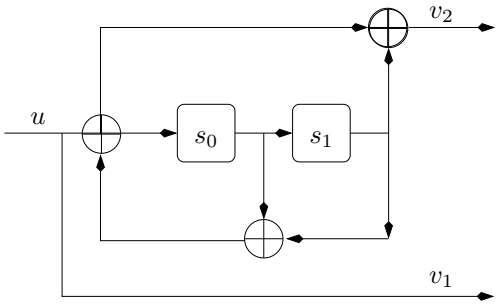
- **Polynomial gen. matrix of a non-systematic feed-forward convolutional encoder**

$$\boldsymbol{G}(D) = (g_1(D), g_2(D)) = \left(1 + D + D^2 \;\; 1 + D^2 \right)$$



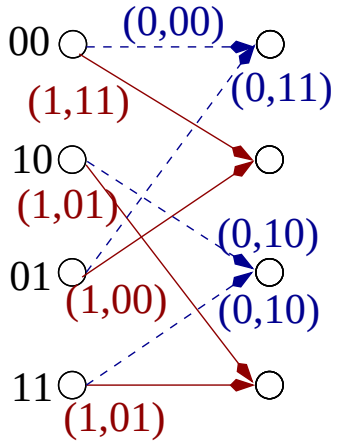
- **Rational generator matrix of the equivalent systematic feedback encoder**

$$\boldsymbol{G}(D) = \left(1, \frac{g_2(D)}{g_1(D)}\right) = \left(1 \;\; \frac{1+D^2}{1+D+D^2} \right)$$

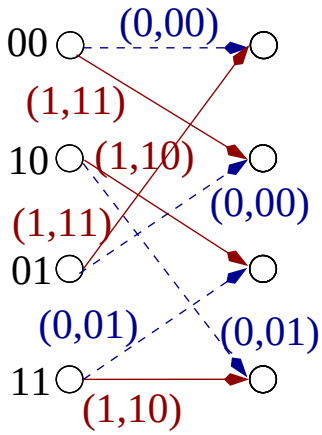


Systematic Feedback Convolutional Encoders (cntd)

$$\boldsymbol{G}(D) = \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \end{pmatrix}$$



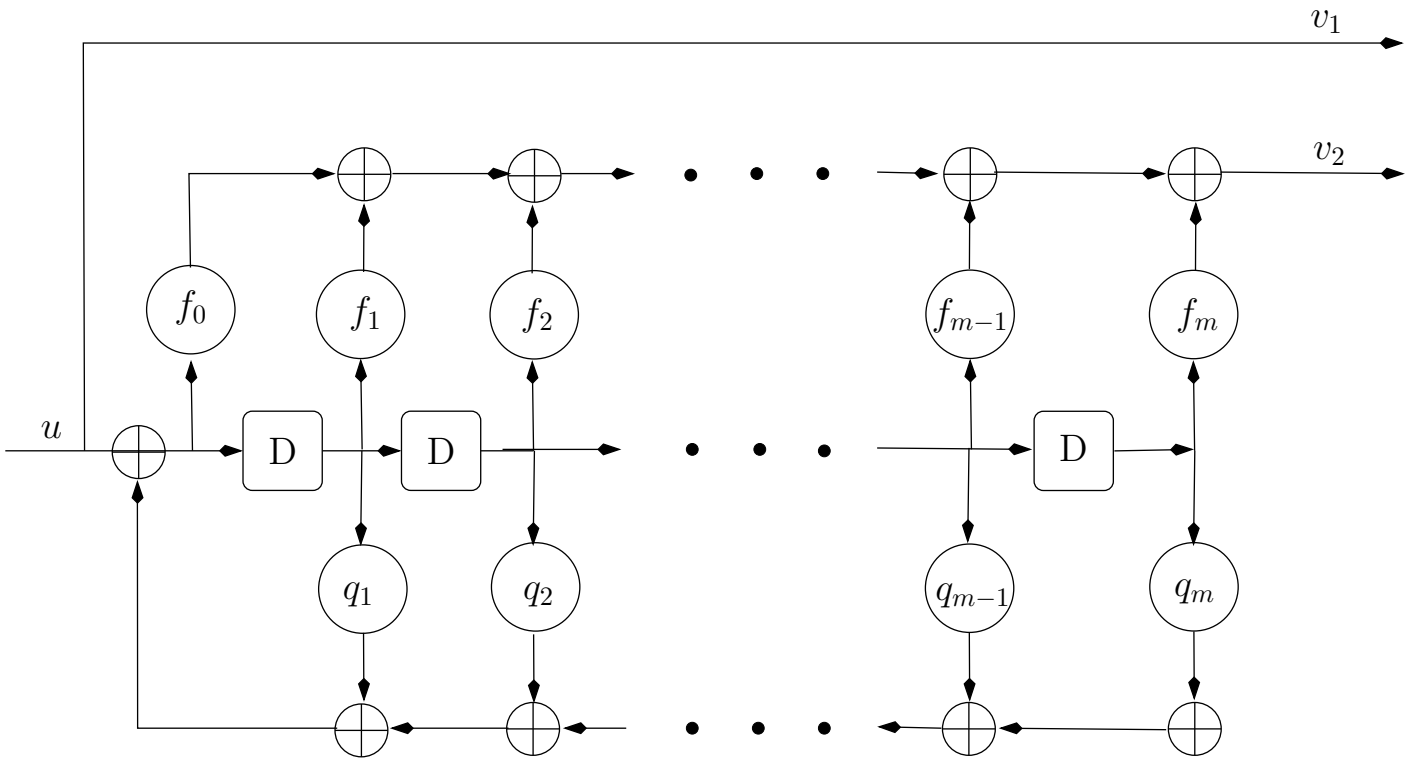
$$\boldsymbol{G}(D) = \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix}$$



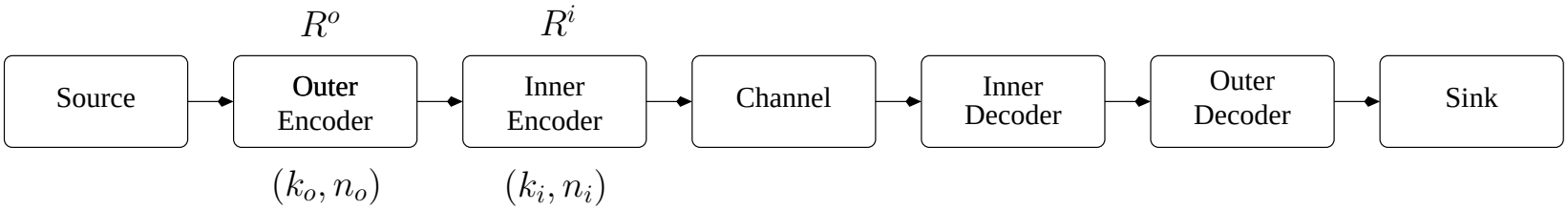
- **Both encoders generate the same code.**
- **The mapping of message bits to coded bit is different.**
- **If we are interested in bit error probability rather than in block error probability, then the choice of the encoder is important.**
- **The choice of the encoder is of central importance in iterative "turbo" decoding.**

General Structure of a Systematic Feedback Encoder

$$\boldsymbol{G}(D) = \left(1 \quad \frac{f_0+f_1D^1...+f_mD^m}{1+q_1D+...+q_mD^m} \right)$$



Concatenated Codes



Forney 1966

Assume

$$n_o = mk_i, \qquad m \text{ integer}$$

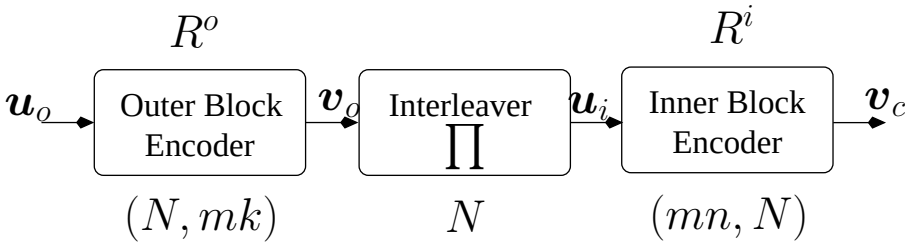
$$R_c = \frac{k_o}{n_o}\frac{n_o}{mn_i} = \frac{k_o}{n_o}\frac{k_i}{n_i} = R^o R^i$$

Properties:

- **Concatenated codes break the burden of decoding the overall (mn_i, k_o) code into a cascade of simpler codes.**
- **Optimum decoding procedure requires a soft output inner decoder, i.e. the inner decoder estimates the a posteriori probability of the coded symbols at the input of the outer decoder.**
- **Typically used with powerful non-binary Reed-Solomon codes and short constrained length convolutional codes as inner codes.**
- **Errors at the Viterbi decoder are bursty while the Reed-Solomon codes assume uncorrelated inputs.**

  $\Rightarrow$ **Correlation can be removed by an interleaver.**

Serially Concatenated Block Codes (SCBC)

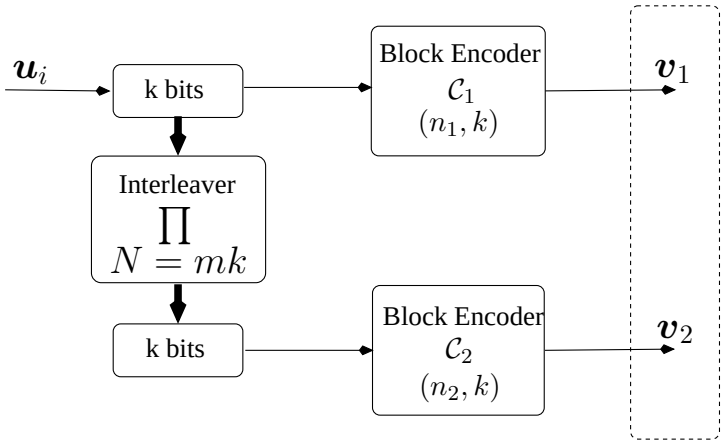


Rate of the outer code: $R^o = \frac{mk}{N}$ **Interleaver length:** N

Rate of the inner code: $R^i = \frac{N}{mn}$ **Rate of the concatenated code:** $R_c^s = \frac{mk}{mn} = R^o R^i$

- **To achieve performance close to the Shannon limit, the information block length mK and the interleaver size N are chosen very large (at least several thousand bits).**
- **The best interleavers reorder bits in a pseudorandom way.**
- **Conventional block (row-column) interleavers do not perform well in turbo codes (except at relatively short block length).**

Parallel Concatenated Block Codes (PCBC)



Block code $\mathcal{C}_1(n_1, k)$ **with rate** $R^{(1)} = \frac{k}{n_1}$

Block code $\mathcal{C}_2(n_2, k)$ **with rate** $R^{(2)} = \frac{k}{n_2}$

Interleaver of size $N = mk$

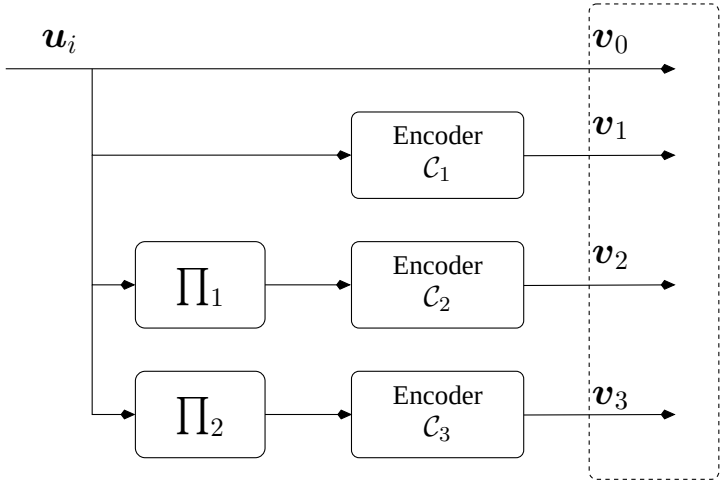
Rate of the concatenated code:

$$R_c^p = \frac{k}{n_1 + n_2} = \frac{k^2}{n_1 n_2 \left(\frac{k}{n_2} + \frac{k}{n_1}\right)} = \frac{R^{(1)} R^{(2)}}{R^{(1)} + R^{(2)}}$$

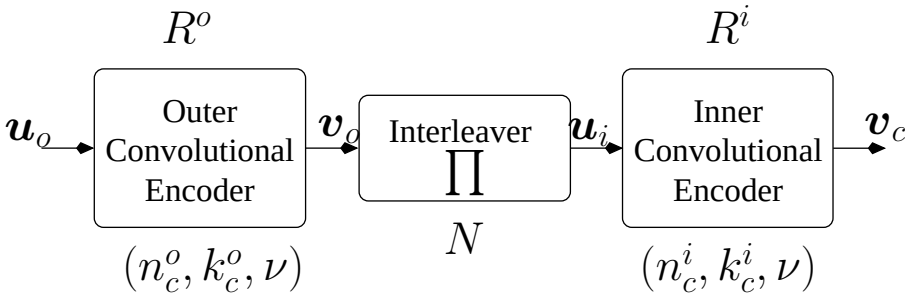
- **The interleaver is an integral part of the overall encoder. Thus the code length of a turbo code is extremely large since the interleaver size is chosen very large and ML and MAP decoding is impossible.**

Parallel Concatenated Block Codes (cntd)

- **Multiple parallel turbo codes are also feasible**



Serial Concatenated Convolutional Codes (SCCC)

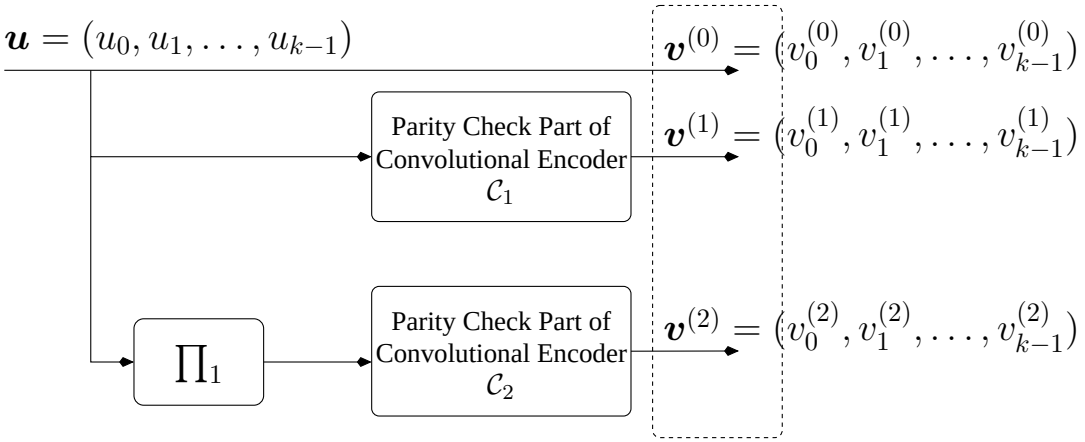


- **Transmission is done on a block-by-block basis: a block of length $K_o = K_o^* + \nu = mk_c^o$ including termination bits enters in the outer convolutional encoder (n_c^o, k_c^o, ν). We assume the interleaving length $N = mn_c^o = m'k_c^i = K_i$ where K_i is the length of the block code entering in the inner convolutional encoder (n_c^i, k_c^i, ν). Note that in this case the trellis of the inner convolutional encoder is not properly terminated, i.e. the inner encoder will not normally return to the 0 state (Modifications can be made to properly terminate the trellis).**
- **Again the rate of the SCCC is $R_s \approx \frac{k_c^o}{n_c^o}\frac{k_c^i}{n_c^i} = R^i R^o$.**
- **After the discovery of turbo codes based on parallel concatenated codes (Berrou et al. 1993) the iterative decoding algorithm was adapted to serially concatenated codes.**
- **Since then, serially concatenated codes have become increasingly popular.**

Serial Concatenated Convolutional Codes (cntd)

- **The best performance at moderate BERs down to about 10^{-5} is achieved with short constrained length constituent encoders, typically $\nu \leq 4$.**
- **Recursive constituent codes generated by systematic feedback encoders give much better performance than feedforward encoders.**

Parallel Concatenated Convolutional Codes (PCCC)



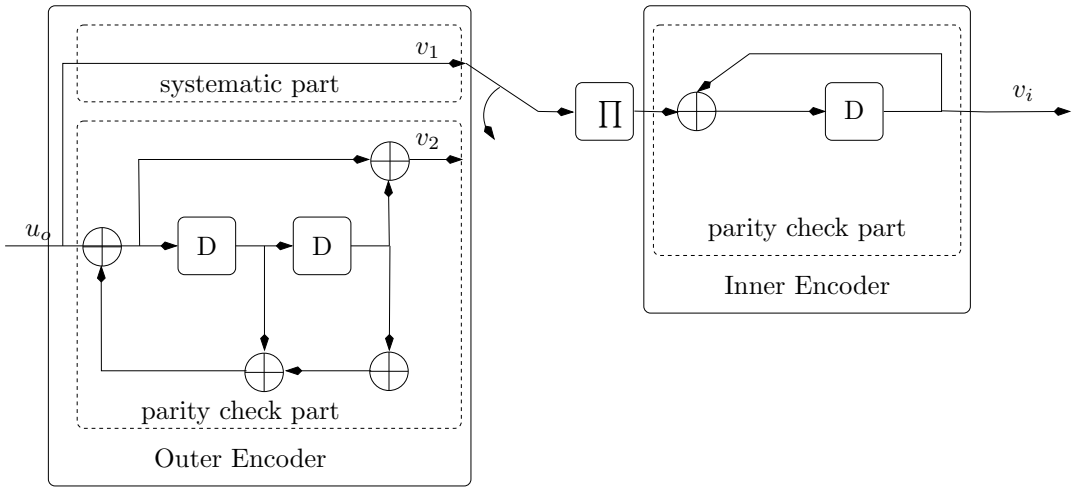
- **The constituent codes are normally generated by the same encoder but this is not a requirement for good performance.**
- **Bit can be punctured from the parity check sequences to produce higher code rates.**
- **Transmission is done on a block-by-block basis. The block length is given by the length of the interleaver. This aspect along with iterative decoding implies long delays. Therefore, turbo codes are not suitable for real time applications such as voice transmission and packet communications in high-speed networks.**

Examples: SCCC

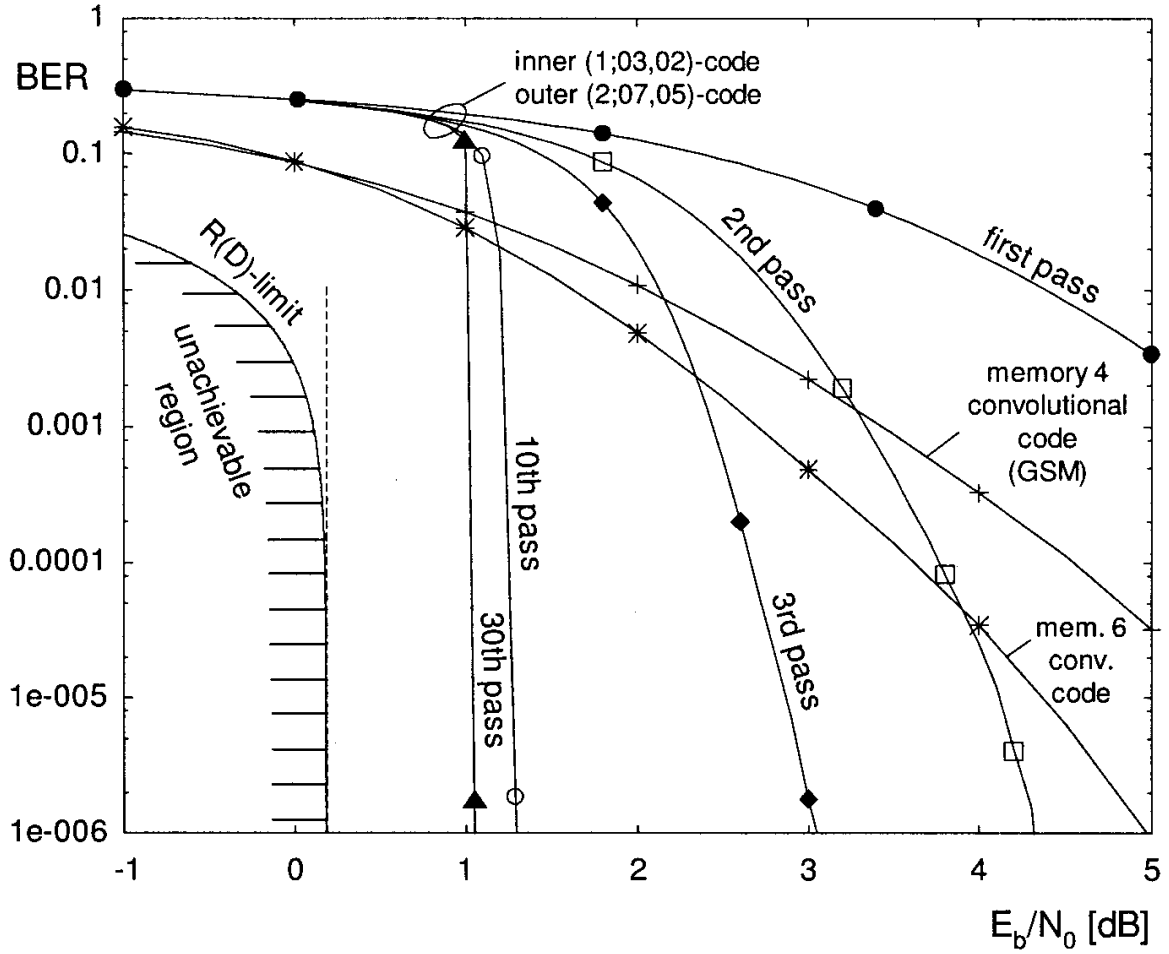
Serial concatenated encoder consisting of

- **an outer recursive systematic convolutional encoder of rate** $R = 1/2$, **memory** $\nu = 2$ **and generating matrix** $\boldsymbol{G} = (07, 05)$
- **an inner differential encoder** $\boldsymbol{G} = (03, 02)$ **with** $R_i = 1$ **and memory** $\nu = 1$.

$$\boldsymbol{G} = (07, 05) = (Z^2 + Z + 1, Z^2 + 1)$$
$$\Rightarrow \left(1, \frac{Z^2 + 1}{Z^2 + Z + 1}\right)$$
$$\boldsymbol{G} = (03, 02) = (Z + 1, Z)$$
$$\Rightarrow \left(1, \frac{Z}{Z + 1}\right)$$



Examples: SCCC (cntd)



Remarks: SCCC (cntd)

- With each iteration, the BER of the serial concatenated scheme is reduced until a saturation point is reached.
- After a few iteration, or "passes", the SC scheme outperforms the convolutional codes.
- The turbo cliff effect is a typical feature of iterative decoding: after exceeding a certain E_b/N_0 value (here $E_b/N_0|_{\text{cliff}} \approx 1.05$ dB) the BER drops by several order of magnitude.
- For short interleaving depth, the turbo cliff is less pronounced.
- We identify three regions of the BER chart:
  - The region of low $E_b/N_0 < E_b/N_0|_{\text{cliff}}$ characterized by negligible iterative BER reduction.
  - The turbo cliff region at about $E_b/N_0 \approx E_b/N_0|_{\text{cliff}}$ with persistent iterative BER reduction over many iterations.
  - The BER floor region for moderate to high E_b/N_0, where a rather low BER can be reached after just a few iterations.

Remarks: SCCC (cntd)

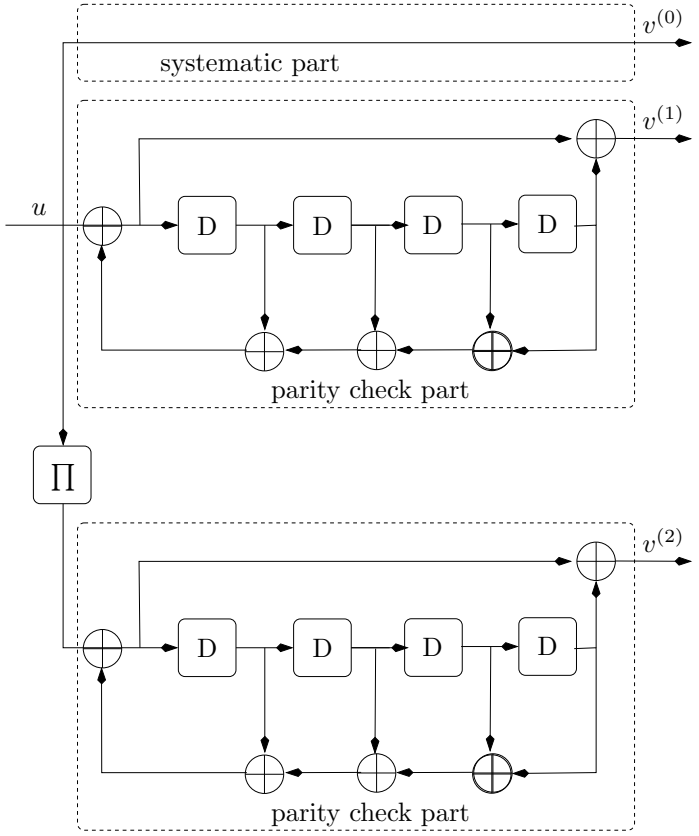
- For $R = 1/2$, the capacity of the BPSK input/continous output Gaussian channel is at $E_b/N_0 \approx 0.19$ dB; thus, this simple serial concatenated code performs within 0.86 dB of capacity.
- In the shown simulations, no residual BER was observed in the BER floor region, although analytical bounds show that the BER is not zero there. The behavior is dominated by the low weight codewords.
- For serially concatenated codes with inner recursive codes, and parallel concatenated codes with recursive component codes: the BER floor can be lowered as much as desired by increasing the interleaving length n_2
- For SC codes the interleaver gain increases as $n_2^{d_{\text{free}}}$, d_{free} even, $n_2^{(d_{\text{free}}+1)/2}$, d_{free} odd, where d_{free} is the free distance of the outer convolutional code.
- For PC the interleaver gain is proportional to the the interleaving depth, and thus to n_2 or n.
- The value $E_b/N_0|_{\text{cliff}}$, however, is a property of a particular code concatenation, i.e. of the component feedforward/feedback polynomials.
- $E_b/N_0|_{\text{cliff}}$ cannot be lowered by increasing the interleaving depth.

Examples: PCCC

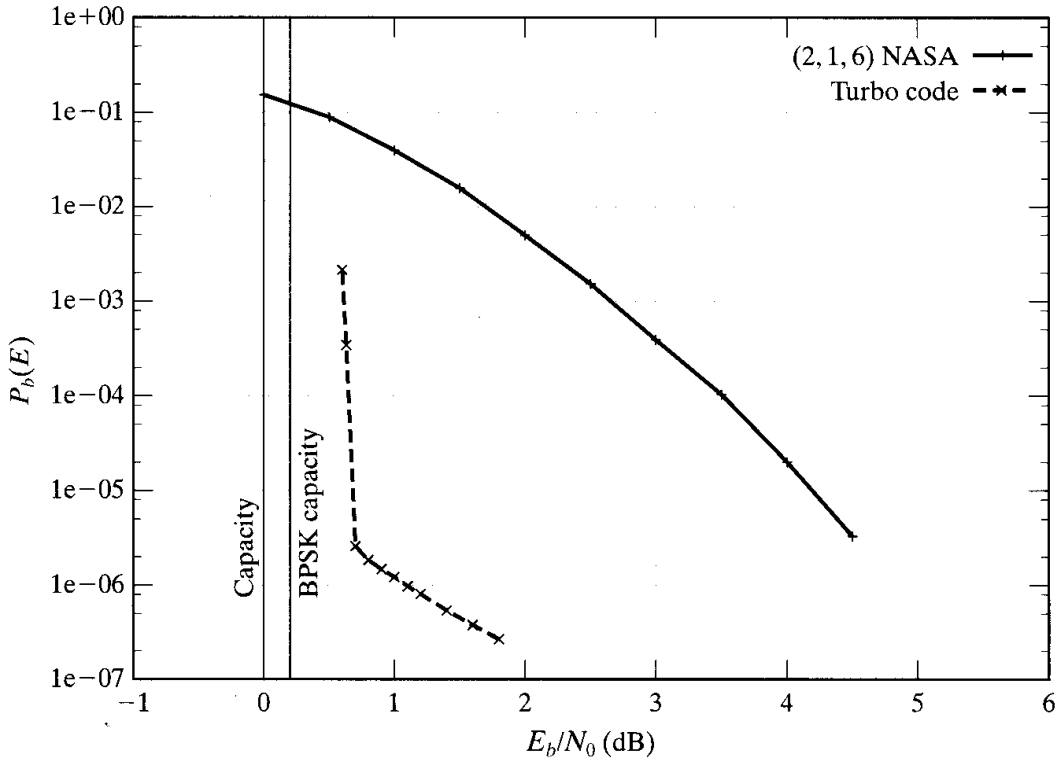
Parallel concatenated encoder consisting of

- **a recursive systematic convolutional encoder with non-systematic feedforward generator matrix** $\boldsymbol{G} = (1 + D + D^2 + D^3 + D^4, 1 + D^4)$ **and rate** $R = 1/2$
- **a second encoder identical to the parity check part of the previous one.**

$$\boldsymbol{G}_{fd} = \left[1, \frac{1 + D^4}{1 + D + D^2 + D^3 + D^4}\right]$$



Examples: PCCC (cntd)



- Turbo codes suffer from significantly weakened performance at BER below 10^{-5} owing to the fact that the codes have a relatively poor minimum distance, which manifests itself at very low BERs. This effect is known as error floor.
- The BER floor can be lowered by a proper design of the interleaver. This can reduce the minimum distance of the code and then improve the performance in the BER floor region.

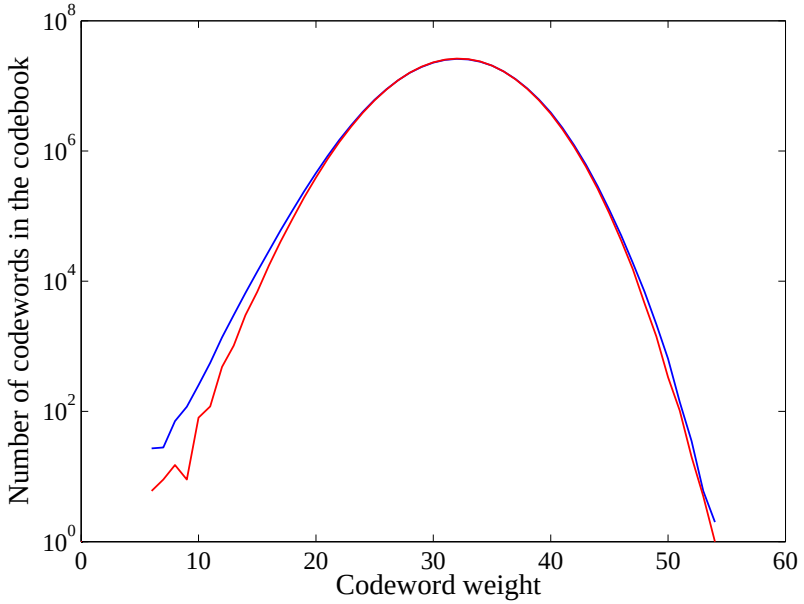
Spectral Thinning

- **Consider a terminated feedforward convolutional encoder with rate $R = 1/2$. If $(001100...)$ is a codeword, because the convolutional encoder is time invariant also $(00001100...), (0000001100...), \ldots$ are codewords. Thus, if a codeword with low weight exists there are also many other codewords with low weight.**
- **Consider a feedback convolutional encoder, it has the same spectrum of the corresponding feedforward encoder**
- **Consider PC feedback systematic encoders with a random interleaver. The minimum distance of the code is substantially unchanged but the number of codewords with low weight decreases substantially. This effect is called spectral thinning and justify the better performance of parallel concatenated codes over convolutional codes.**

Spectral Thinning: Example

Consider the example of the PCCC in slide 18 and assume an interleaving of depth 28, then the spectra of the convolutional code and of the PCCC is shown in the following table.

(a) Terminated convolutional				(b) Parallel concatenated			
Weight	Multiplicity	Weight	Multiplicity	Weight	Multiplicity	Weight	Multiplicity
0	1	33	25431436	0	1	33	25716960
1	0	34	23509909	1	0	34	23669905
2	0	35	20436392	2	0	35	20464409
3	0	36	16674749	3	0	36	16623260
4	0	37	12757248	4	0	37	12662360
5	0	38	9168248	5	0	38	9024333
6	27	39	6179244	6	6	39	6012086
7	28	40	3888210	7	9	40	3729485
8	71	41	2271250	8	15	41	2156481
9	118	42	1226350	9	9	42	1160459
10	253	43	615942	10	80	43	573214
11	558	44	287487	11	119	44	262676
12	1372	45	124728	12	484	45	110369
13	3028	46	50466	13	1027	46	42264
14	6573	47	19092	14	3007	47	15269
15	14036	48	6888	15	6852	48	4556
16	29171	49	2172	16	17408	49	1416
17	60664	50	642	17	40616	50	335
18	122093	51	140	18	90244	51	103
19	240636	52	35	19	193196	52	20
20	457660	53	6	20	390392	53	5
21	838810	54	2	21	754819	54	1
22	1476615	55	0	22	1368864	55	0
23	2484952	56	0	23	2367949	56	0
24	3991923	57	0	24	3874836	57	0
25	6098296	58	0	25	5988326	58	0
26	8845265	59	0	26	8778945	59	0
27	12167068	60	0	27	12149055	60	0
28	15844169	61	0	28	15907872	61	0
29	19504724	62	0	29	19684668	62	0
30	22702421	63	0	30	22978613	63	0
31	24967160	64	0	31	25318411	64	0
32	25927128			32	26289667		



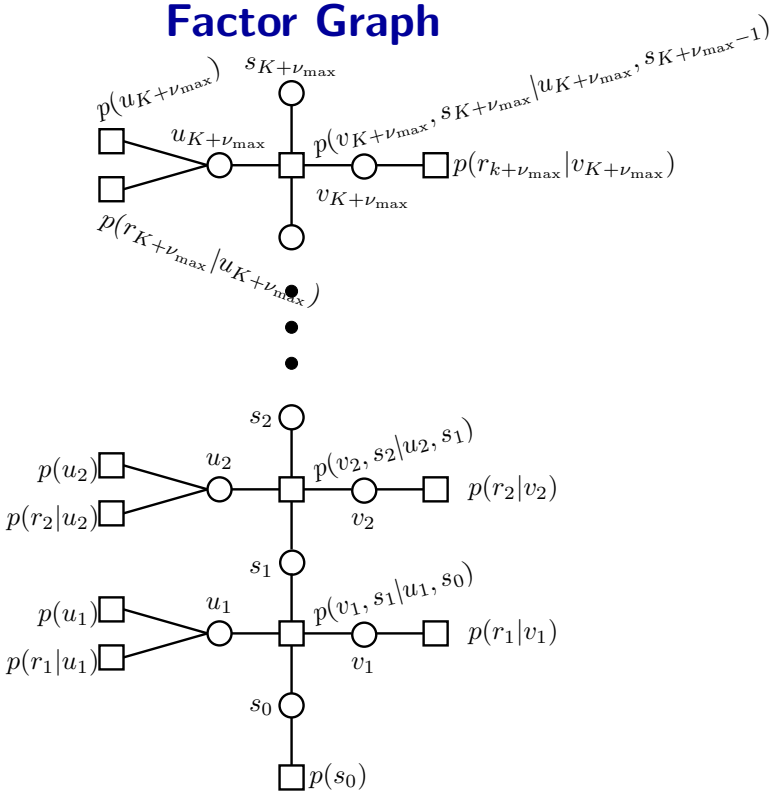
BCJR Algorithm Revisited

- **Systematic convolutional code with rate $1/2$ (one information bit u_ℓ and a parity check bit v_ℓ)**
- **$\boldsymbol{r}^s$ received symbols corresponding to transmitted information bits.**
- **$\boldsymbol{r}^p$ received symbols corresponding to transmitted parity check bits.**

MAP for Convolutional Codes

$$
\begin{aligned}
\widehat{u}_\ell &= \underset{u_\ell \in \{0,1\}}{\operatorname{argmax}} p(u_\ell | \boldsymbol{r}^s, \boldsymbol{r}^p) \\
&= \underset{u_\ell \in \{0,1\}}{\operatorname{argmax}} \sum_{\sim u_\ell} p(\boldsymbol{u}, \boldsymbol{v}^p, \boldsymbol{s} | \boldsymbol{r}^s, \boldsymbol{r}^p) \\
&= \underset{u_\ell \in \{0,1\}}{\operatorname{argmax}} \sum_{\sim u_\ell} p(\boldsymbol{r}^s, \boldsymbol{r}^p | \boldsymbol{u}, \boldsymbol{v}^p, \boldsymbol{s}) p(\boldsymbol{u}, \boldsymbol{v}^p, \boldsymbol{s}) \\
&= \underset{u_\ell \in \{0,1\}}{\operatorname{argmax}} \sum_{\sim u_\ell} p(s_0) \prod_{j=1}^{K+\nu_{MAX}} p(r_j^s | u_j) p(r_j^p | v_j^p) \\
&\quad \times p(u_j) p(v_j^p, s_j | u_j, s_{j-1})
\end{aligned}
$$

Factor Graph



BCJR Algorithm Revisited: Remarks

- **The function $p(v_j, s_j|u_j, s_{j-1})$ is a function with values on the set $\{0, 1\}$. It is usually represented by a trellis.**
- **The factor graph does not contain cycles. Then, the message passing algorithm provides the exact marginal pdf.**
- **The message passing algorithms is over a line. In order to determine all the marginal pdfs, each edge needs to be considered in both directions.**
- **The message passing from the bottom to the top is the so-called α iteration.**
- **The message passing from the top to the bottom is the so-called β iteration.**
- **The final step for the computation of $p(u_\ell|\boldsymbol{r})$is called γ step in literature.**
- **The BCJR algorithm corresponds to a sum product message passing algorithm.**
- **The step compare and add of the Viterbi algorithm correspond to the max-sum step of a max-sum message passing algorithm. Then, the max-sum message passing over the factor graph of a convolutional code is equivalent to the Viterbi algorithm.**

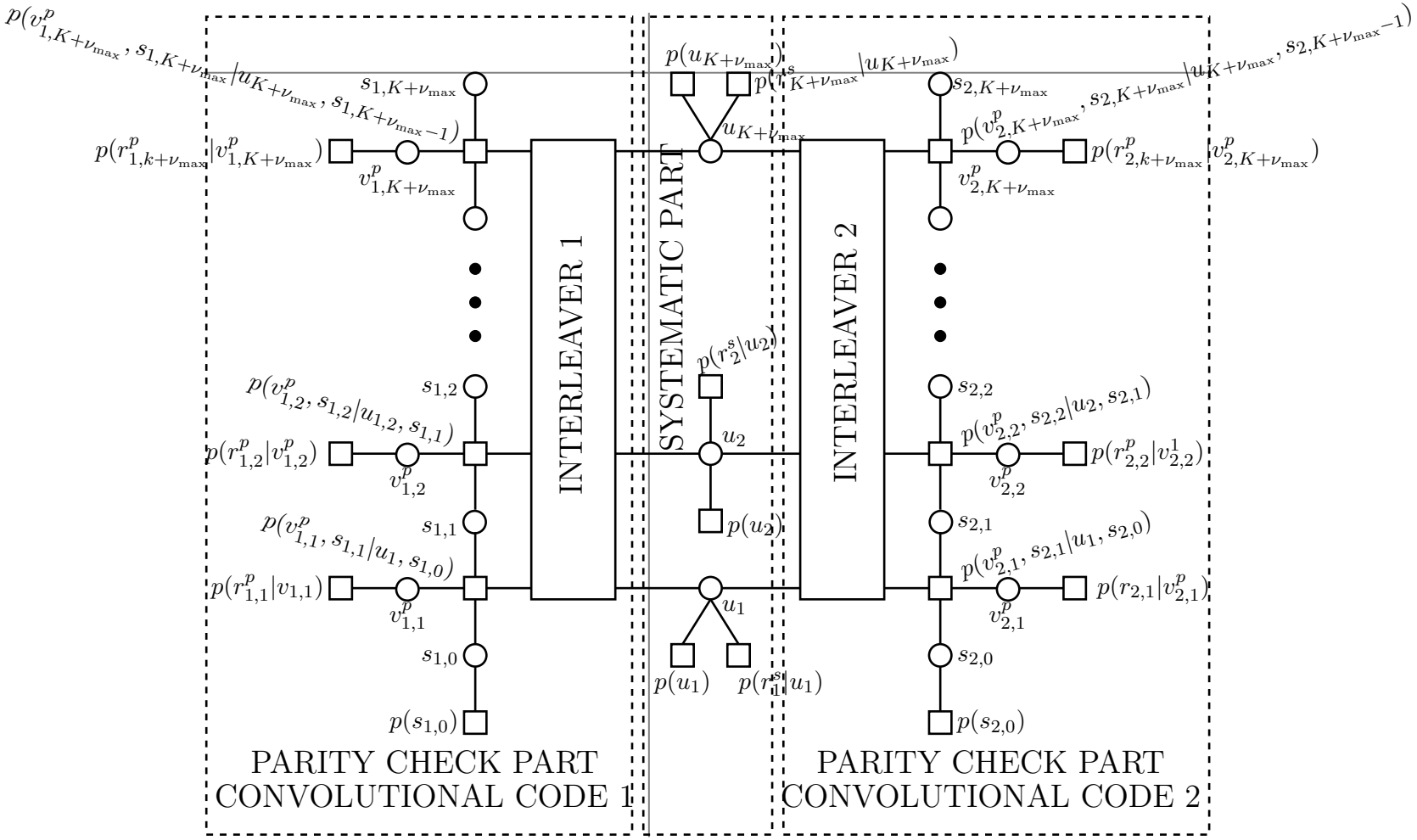
Factor Graph for Turbo Decoding

Assume

- **Two parallel concatenated systematic convolutional codes with rate 1/2, called code 1 and code 2.**
- **The output of the encoder is $\boldsymbol{v}_\ell = (u_\ell, v^p_{1,\ell}, v^p_{2,\ell})$, where u_ℓ is the systematic component, $v^p_{1,\ell}$ and $v^p_{1,\ell}$ are the parity check components for the first and second encoder, respectively.**
- **The output of the systematic part is the vector $\boldsymbol{u}$, the output of the parity check parts are the two vectors $\boldsymbol{v}^p_1$ and $\boldsymbol{v}^p_2$ for code 1 and code 2 respectively.**
- **The vectors $\boldsymbol{r}^s, \boldsymbol{r}^p_1$, and $\boldsymbol{r}^p_2$ are the output of the memoryless channel to the inputs $\boldsymbol{u}, \boldsymbol{v}^p_1$, and $\boldsymbol{v}^p_2$, respectively.**
- **The vectors $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are the sequence of the states of the convolutional codes 1 and 2, respectively, when the message $\boldsymbol{u}$ is transmitted.**

The factor graph for turbo decoding is obtained by properly combining the factor graphs of the BCJR algorithms for the two convolutional codes

Factor Graph for Decoding a PCCC with Interleaving



A-Priori, Channel, Extrinsic L-Values: an Example

- **Consider a block code $\mathcal{C} = (2, 4)$ with message bits $\boldsymbol{u} = (u_0, u_1)$ and received coded bits $\boldsymbol{r} = (r_0, r_1, r_2, r_3)$.**
- **Assume the message bits u_0, u_1 independent, i.e. $P(u_0, u_1) = P(u_0)P(u_1)$.**
- **Compute the two APP L-values on the message bits u_0, u_1.**
- **The APP L-value of u_0 conditioned on $\boldsymbol{r}$ is**

$$L_D(u_0|\boldsymbol{r}) = \ln \frac{P(u_0 = 0|\boldsymbol{r})}{P(u_0 = 1|\boldsymbol{r})} = \ln \frac{P(u_0 = 0, \boldsymbol{r})}{P(u_0 = 1, \boldsymbol{r})}$$

- **Note that**

$$\begin{aligned} P(u_0 = 0, \boldsymbol{r}) &= P(u_0 = 0, u_1 = 0, \boldsymbol{r}) + P(u_0 = 0, u_1 = 1, \boldsymbol{r}) \\ &= P(u_0 = 0)P(u_1 = 0)P(\boldsymbol{r}|u_0 = 0, u_1 = 0) \\ &+ P(u_0 = 0)P(u_1 = 1)P(\boldsymbol{r}|u_0 = 0, u_1 = 1). \end{aligned}$$

A-Priori, Channel, Extrinsic L-Values: an Example (condt)

- **We obtain**

$$
\begin{aligned}
L_D(u_0|\boldsymbol{r}) &= \ln \frac{P(\boldsymbol{r}|u_0=0,u_1=0)P(u_0=0)P(u_1=0)+P(\boldsymbol{r}|u_0=0,u_1=1)P(u_0=0)P(u_1=1)}{P(\boldsymbol{r}|u_0=1,u_1=0)P(u_0=1)P(u_1=0)+P(\boldsymbol{r}|u_0=1,u_1=1)P(u_0=1)P(u_1=1)} \\
&= \ln \frac{P(u_0=0)}{P(u_0=1)} + \ln \frac{P(\boldsymbol{r}|u_0=0,u_1=0)\frac{P(u_1=0)}{P(u_1=1)}+P(\boldsymbol{r}|u_0=0,u_1=1)}{P(\boldsymbol{r}|u_0=1,u_1=0)\frac{P(u_1=0)}{P(u_1=1)}+P(\boldsymbol{r}|u_0=1,u_1=1)} \\
&= L_A(u_0) + \underbrace{\ln \frac{P(\boldsymbol{r}|u_0=0,u_1=0)\exp(L_A(u_1))+P(\boldsymbol{r}|u_0=0,u_1=1)}{P(\boldsymbol{r}|u_0=1,u_1=0)\exp(L_A(u_1))+P(\boldsymbol{r}|u_0=1,u_1=1)}}_{L_{E'}(u_0|\boldsymbol{r})}
\end{aligned}
$$

- **The APP L-value $L_D(u_0|\boldsymbol{r})$ is the sum of the a priori L-value $L_A(u_0)$ and the channel and extrinsic L-value $L_{E'}(u_0|\boldsymbol{r})$.**
- **The hard decision on u_0 is $\mathrm{sign} L_D(u_0|\boldsymbol{r})$. $|L_D(u_0|\boldsymbol{r})|$ gives the reliability of the decision.**

A-Priori, Channel, Extrinsic L-Values: Systematic Codes

- **Consider a systematic block code $\mathcal{C} = (2, 4)$ with message bits $\boldsymbol{u} = (u_0, u_1)$, coded bits $\boldsymbol{v} = (v_0 = u_0, v_1 = u_1, v_2, v_3)$, and received coded bits $\boldsymbol{r} = (r_0, r_1, r_2, r_3)$.**
- **Say $\boldsymbol{r}_{\sim i} = (r_0, r_1 \ldots, r_{j-1}, r_{j+1} \ldots r_{N-1})$**
- **For memoryless channels we can write**

$$P(\boldsymbol{r}|\boldsymbol{u}) = P(\boldsymbol{r}|\boldsymbol{v}(\boldsymbol{u})) = P(r_i|v_i(\boldsymbol{u})) \cdot P(\boldsymbol{r}_{\sim i}|\boldsymbol{v}_{\sim i}(\boldsymbol{u}))$$

and therefore

$$L_D(u_0|\boldsymbol{r}) = L_A(u_0) + \underbrace{\ln \frac{P(r_0|u_0 = 0)}{P(r_0|u_0 = 1)}}_{L_{\text{ch}}(u_0|r_0)} + \underbrace{\ln \frac{P(\boldsymbol{r}_{\sim 0}|u_0 = 0, u_1 = 0)\exp(L_A(u_1)) + P(\boldsymbol{r}_{\sim 0}|u_0 = 0, u_1 = 1)}{P(\boldsymbol{r}_{\sim 0}|u_0 = 1, u_1 = 0)\exp(L_A(u_1)) + P(\boldsymbol{r}_{\sim 0}|u_0 = 1, u_1 = 1)}}_{L_E(u_0|\boldsymbol{r}_{\sim 0})}$$

- **The APP L-value $L_D(u_0|\boldsymbol{r})$ is the sum of the a priori L-value $L_A(u_0)$ and the channel L-value $L_{\text{ch}}(u_0|r_0)$ and the extrinsic L-value $L_E(u_0|\boldsymbol{r})$.**

A-Priori, Channel, Extrinsic L-Values: General Expressions

- **Consider a block code $\mathcal{C} = (K, N)$ with message bits $\boldsymbol{u} = (u_0, \ldots, u_{K-1})$, coded bits $\boldsymbol{v}$, and received coded bits $\boldsymbol{r}$.**
- **We have**

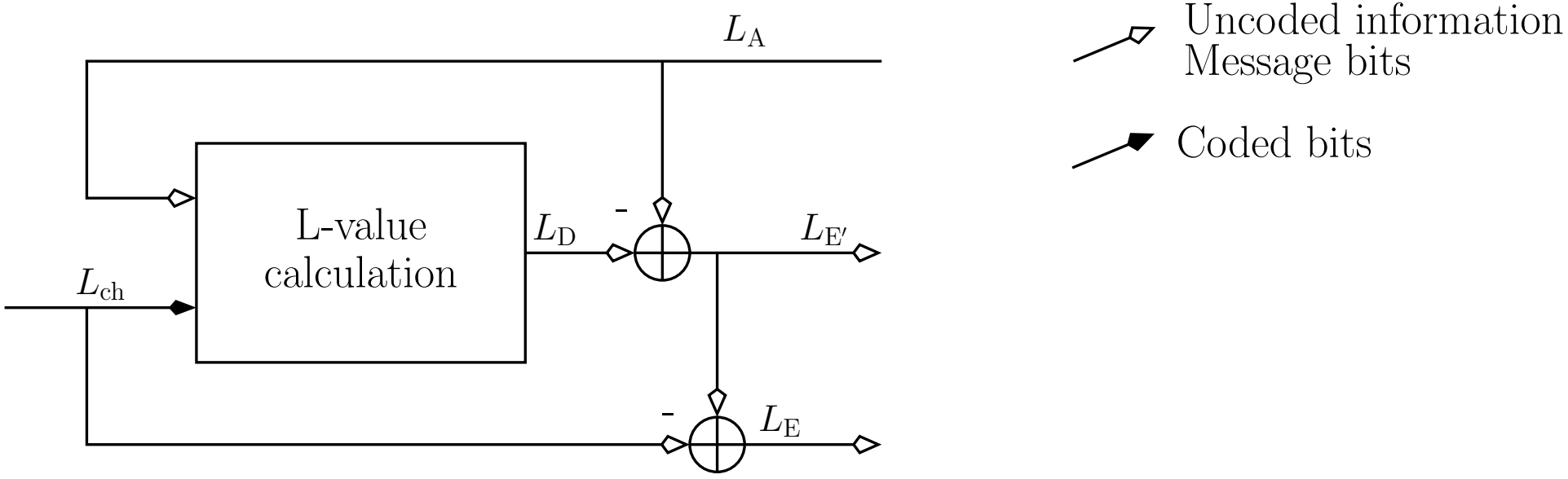
$$L_D(u_i|\boldsymbol{r}) = L_A(u_i) + \underbrace{\ln \frac{\sum_{\boldsymbol{u}:\, u_i=0} P(\boldsymbol{r}|\boldsymbol{u}) \exp\left[\sum_{j:j\neq i,\, u_j=0} L_A(u_j)\right]}{\sum_{\boldsymbol{u}:\, u_i=1} P(\boldsymbol{r}|\boldsymbol{u}) \exp\left[\sum_{j:j\neq i,\, u_j=0} L_A(u_j)\right]}}_{L_{E'}(u_0|\boldsymbol{r})}$$

- **For memoryless channels and systematic codes with $\boldsymbol{v} = (v_0 = u_0, \ldots v_{K-1} = u_{K-1}, v_K \ldots, v_{N-1})$, we can extract further the pure extrinsic information for u_i, $0 \leq i < K$**

$$L_D(u_i|\boldsymbol{r}) = L_A(u_i) + L_{\text{ch}}(u_i|r_i) + \underbrace{\ln \frac{\sum_{\boldsymbol{u}:\, u_i=0} P(\boldsymbol{r}_{\sim i}|\boldsymbol{u}) \exp\left[\sum_{j:j\neq i,\, u_j=0} L_A(u_j)\right]}{\sum_{\boldsymbol{u}:\, u_i=1} P(\boldsymbol{r}_{\sim i}|\boldsymbol{u}) \exp\left[\sum_{j:j\neq i,\, u_j=0} L_A(u_j)\right]}}_{L_E(u_i|\boldsymbol{r}_{\sim i})}$$

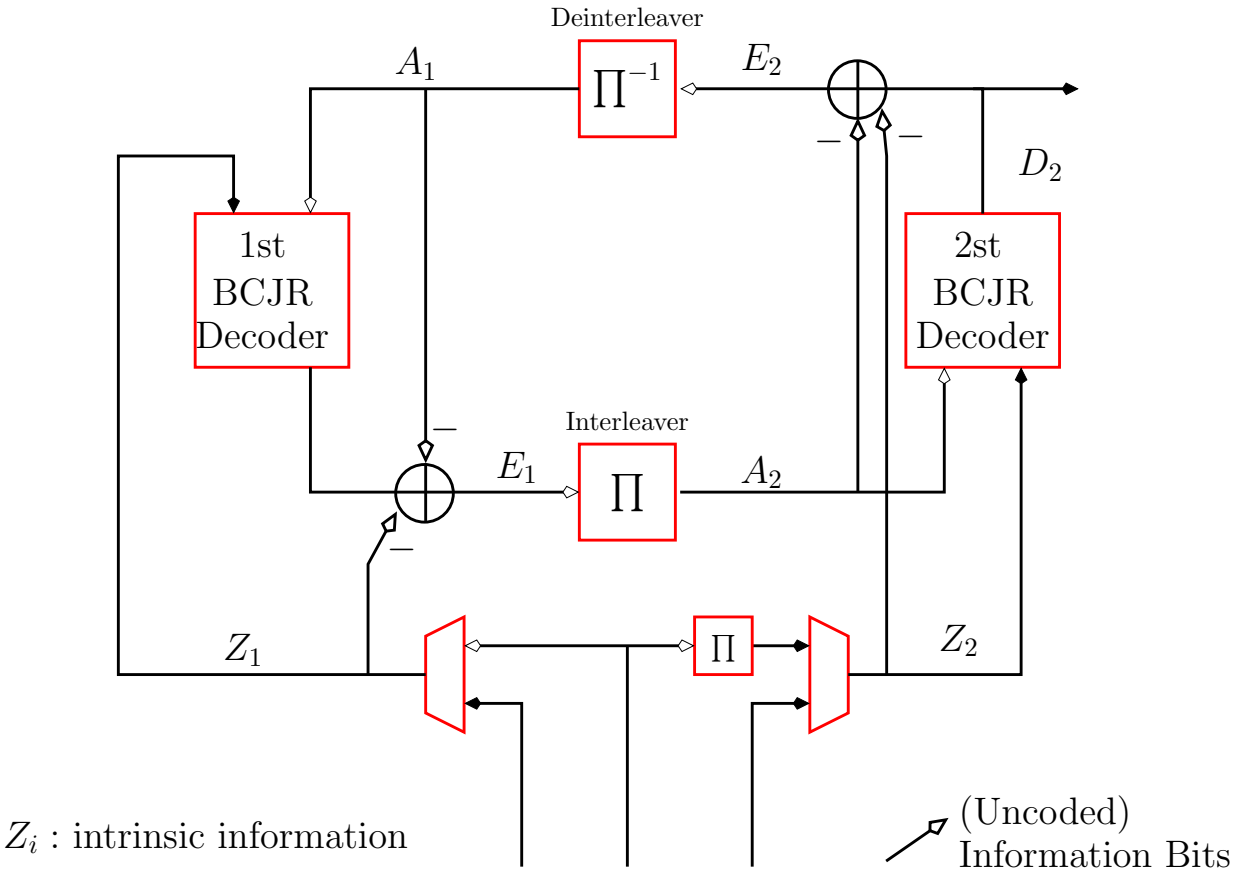
Soft Output Decoding of Message (Info) Bits

Graphical Overview of L-values at the APP decoder



It is typically used for soft output decoding of parallel concatenated codes, or of inner codes in the context of serially concatenated codes.

Turbo Decoding for PCCC



Z_i : intrinsic information

A_i : a priori information

D_i : a posteriori information

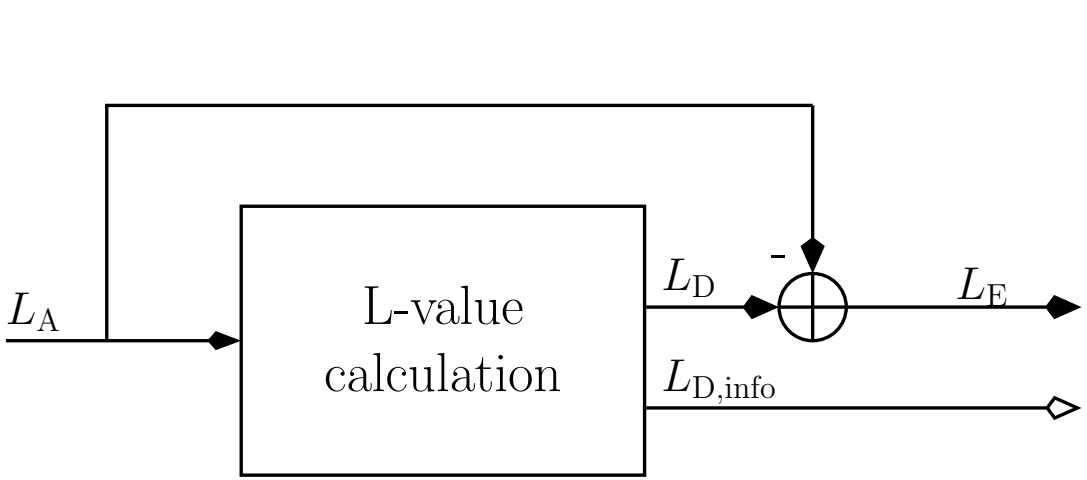
E_i : estrinsic information $(D_i - A_i - Z_i)$

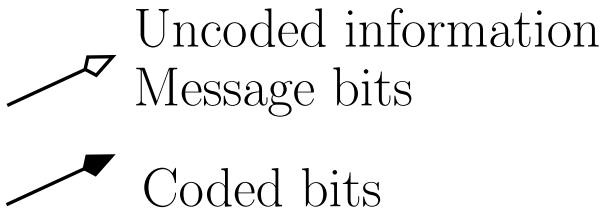
Turbo Decoding for PCCC (cntd)

- **For each iteration, the first component decoder takes intrinsic information (channel observations) Z_1 on the systematic bits u and parity check bits p_1 and computes soft values D_1 (a posteriori log-likelihood ratios) by a BCJR decoder.**
- **The extrinsic information on systematic bits is computed as $E_1 = D_1 - A_1 - Z_1$ and it is passed through the interleaver to become the a priori knowledge A_2 of the second decoder.**
- **The second decoder takes the permuted channel observations B_2 on the systematic bits u and the parity check bits p_2 from the second decoder and feed-back extrinsic information $E_2 = D_2 - A_2 - Z_2$ that becomes the a priori knowledge A_1 of the first decoder.**
- **The variables $Z_1, A_1, D_1, E_1, Z_2, A_2, D_2, E_2$ are log-likelihood ratios.**

Soft Output Decoding of Codeword (coded) Bits

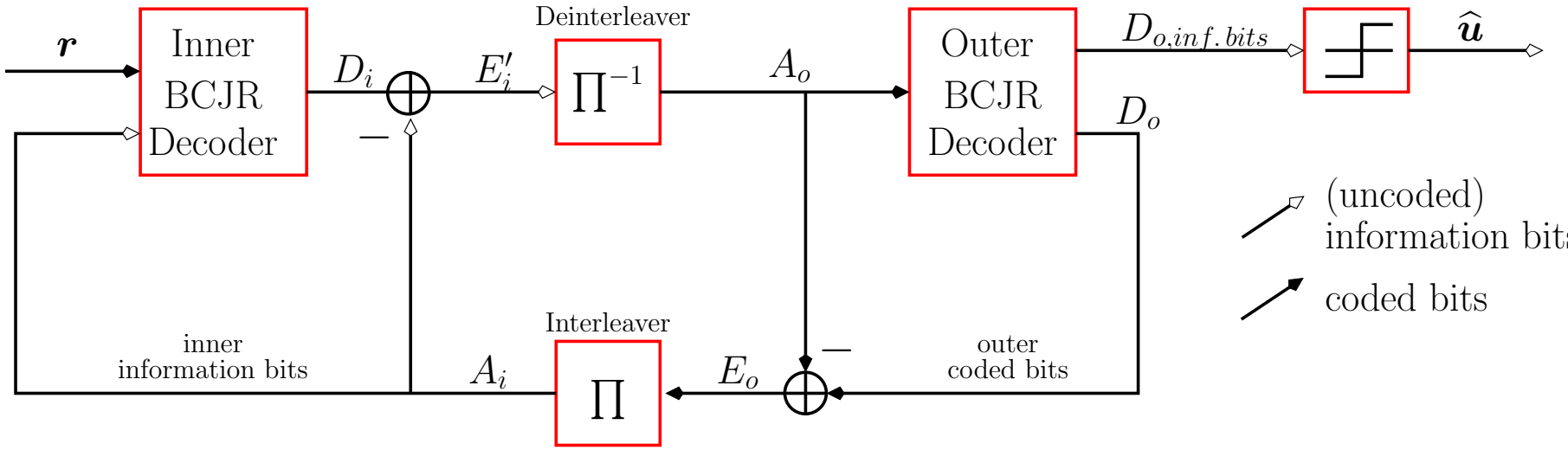
Graphical Overview of L-values at the APP decoder





It is typically used for soft output decoding of outer codes in the context of serially concatenated codes.

Turbo Decoding for SCCC



$D_{\{i,o\}}$: a posteriori information
$A_{\{i,o\}}$: a priori information
$Z_{\{i,o\}}$: intrinsic information

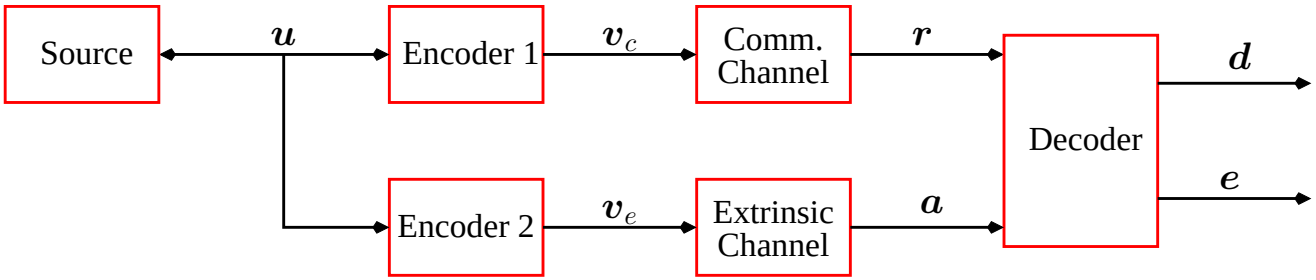
$E_{\{i,o\}}$: estrinsic information $(D_{\{i,o\}} - A_{\{i,o\}} - Z_{\{i,o\}})$
$E'_{\{i,o\}} = D_{\{i,o\}} - A_{\{i,o\}}$

Turbo Decoding for SCCC (cntd)

- **The inner decoder uses the channel observations r to compute a posteriori probabilities (APP) log-likelihood ratios D_i that are passed on as channel and extrinsic information $E'_i = D_i - A_i$ through a bit deinterleaver to become the outer a priori knowledge A_o of the outer decoder.**
- **At the first pass through the inner decoder, the a priori log-likelihood values A_i are zero.**
- **The outer decoder computes the APP log-likelihood ratios D_o on the outer coded bits, and the D_o are fed back as extrinsic information $E_o = D_o - A_o$ to become the a priori knowledge A_i on the inner information bits.**
- **The extrinsic information E_o or a priori knowledge A_1 represents information gained by exploiting the outer code constraints.**
- **The variables $A_i, D_i, E'_i, A_o, D_o, E_o$ are log-likelihood ratios.**

Performance Analysis at High SNR

Single Component Decoder: Decoding Model

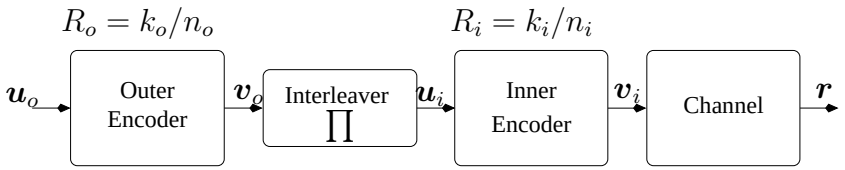


Operation of general component decoders can be modelled as above:

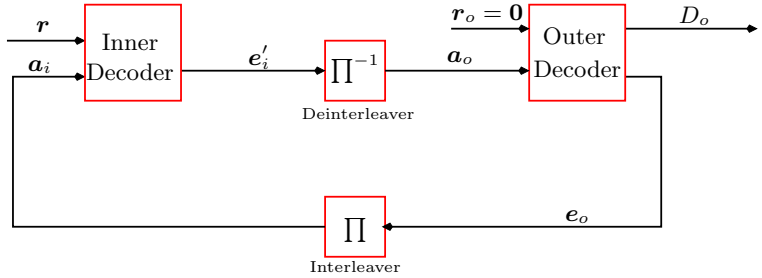
- **The chain $u - v_c - r$ represents the information flow through the actual channel, called the communication channel.**
- **The chain $u - v_e - a$ represents the information flow inside the iterative decoder. The extrinsic channel (or a priori channel) models the statistics of the extrinsic messages being passed.**
- **The vectors a,e,d are log-likelihood ratios. We write m for their length.**

Decoding Model for SC Codes

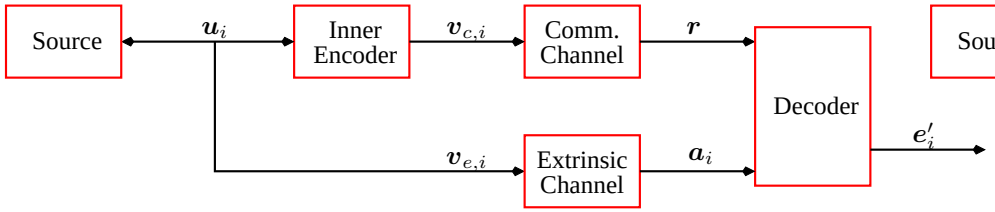
Encoder:



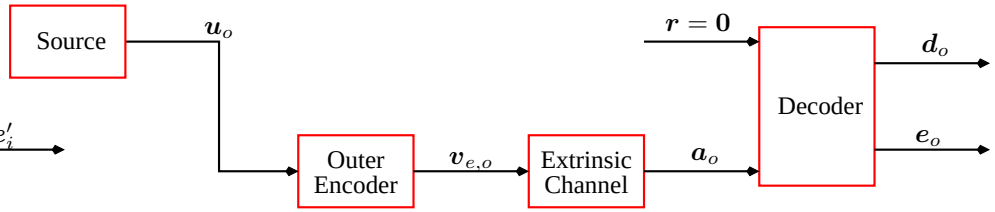
Decoder:



Inner Decoding Model



Outer Decoding Model



Iterative Decoding Analysis

- **The communication channel is usually fixed. In contrast, the extrinsic channel changes from iteration to iteration.**
- **Gallager[1] suggested "to calculate the probability distributions of log-likelihood ratios...for a number of iterations".**
- **Richardson and Urbanke[1] called this density evolution, showed how to simplify the calculation, and discovered several theoretical results on the coding convergence.**
- **Problem: the numerical density evolution analysis is complicated because one tracks density functions.**

[1] R.G. Gallager, "Low density parity check codes", MIT Press, 1963.

[2] T.J. Richardson and R.L. Urbanke, "The capacity of Low-Density-Parity-Check Codes under message passing decoding," IEEE Trans. Information Theory, Vol. IT-47, pp. 619–637, Feb. 2001.

Iterative Decoding Analysis (cntd)

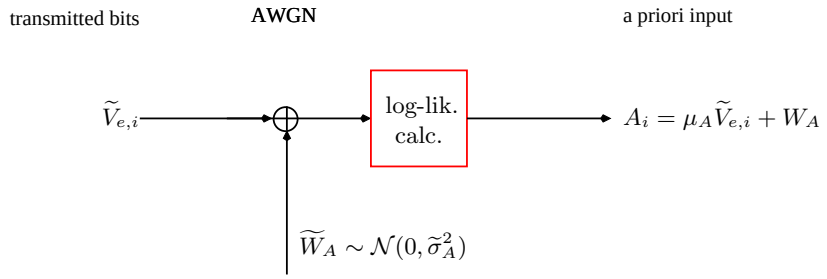
- **Simplification 1: Track only a statistic of the density such as mean, variance, error probability or mutual information.**
  We use the mutual information because
  - **it seems the most accurate and robust statistic**
  - **it applies to discrete and continuous channels**
  - **it has provable properties.**
- **Simplification 2: model the extrinsic channel as memoryless, time invariant and Gaussian.**
  - **This model lets one characterize the component decoders individually**
  - **the memoryless and time-invariant hypotheses can be made accurate by choosing "good" long interleavers**
  - **the Gaussian hypothesis is empirically accurate.**

EXIT Functions

- **EXIT functions characterize how the decoder converts knowledge about v_e, the coded bits in the extrinsic channel, from a to v_e.**
- **Knowledge at a :** $I_A = \frac{1}{m}\sum_{j=1}^{m} I(V_{e,j}, A_j) = I(V_{e,1}, A_1)$.
- **Knowledge at e :** $I_E = \frac{1}{m}\sum_{j=1}^{m} I(V_{e,j}, E_j)$.
- **Consider that $I_E(I_A, E_b/N_0)$. For a given value of E_b/N_0 we obtain the EXIT function $I_E(I_A)$.**

EXIT For the Inner Decoder: Computing I_{A_i}

- **Simulations with iterative decoders show that the extrinsic log-likelihood values E_o fed back from the outer decoder (or the a priori $A = A_i$ at the inner decoder) are almost Gaussian distributed**



- **We model the a priori input A by adding to $\widetilde{V}_{e,i}$ an independent Gaussian random variable $\widetilde{W}_A$ with zero mean and variance $\widetilde{\sigma}_A^2$ where $\widetilde{V}_{e,i} = 1 - 2V_{e,i}$ so that $\widetilde{V}_{e,i} \in \{\pm 1\}$.**
- **We have**

$$A = \frac{2}{\widetilde{\sigma}_A^2}(\widetilde{V}_{e,i} + \widetilde{W}_A) = \mu_A \widetilde{V}_{e,i} + W_A$$

where $\mu_A = \frac{2}{\widetilde{\sigma}_A^2}$ and $\sigma_A^2 = \frac{4}{\widetilde{\sigma}_A^2}$. Note that $\mu_A^2 = \frac{\sigma_A^2}{2}$.

EXIT For the Inner Decoder: Computing I_{A_i} (cndt)

- **The distribution of A conditioned on $\widetilde{V}_{e,i} = \widetilde{v}_{e,i}$ is thus**

$$p_A(\xi|\widetilde{V}_{e,i} = \widetilde{v}_{e,i}) = \frac{\mathrm{e}^{-\frac{(\xi_{e,i} - \frac{\sigma_A^2}{2}\widetilde{v})^2}{2\sigma_A^2}}}{\sqrt{2\pi}\sigma_A}$$

- **We expand $I_A = I(V_{i,e}; A)$ to**

$$I_A = \frac{1}{2} \sum_{\widetilde{v}_{e,i}=\{\pm 1\}} \int_{-\infty}^{+\infty} p_A(\xi|\widetilde{V}_{e,i} = \widetilde{v}_{e,i}) \log_2 \frac{2p_A(\xi|\widetilde{V}_{e,i} = \widetilde{v}_{e,i})}{p_A(\xi|\widetilde{V}_{e,i} = -1) + p_A(\xi|\widetilde{V}_{e,i} = +1)} \mathrm{d}\xi$$

which can also be written as

$$I_A(\sigma_A^2) = 1 - \int_{-\infty}^{+\infty} \frac{\mathrm{e}^{-\frac{(\xi_{e,i} - \frac{\sigma_A^2}{2})^2}{2\sigma_A^2}}}{\sqrt{2\pi}\sigma_A} \log_2(1 + \mathrm{e}^{-\xi}) \mathrm{d}\xi$$

EXIT For the Inner Decoder: Computing I_{A_i} (cndt)

- **We will write**

$$J(\sigma) \stackrel{\triangle}{=} I_A(\sigma_A = \sigma)$$

Note that

$$\lim_{\sigma \to 0} J(\sigma) = 0, \quad \lim_{\sigma \to +\infty} J(\sigma) = 1$$

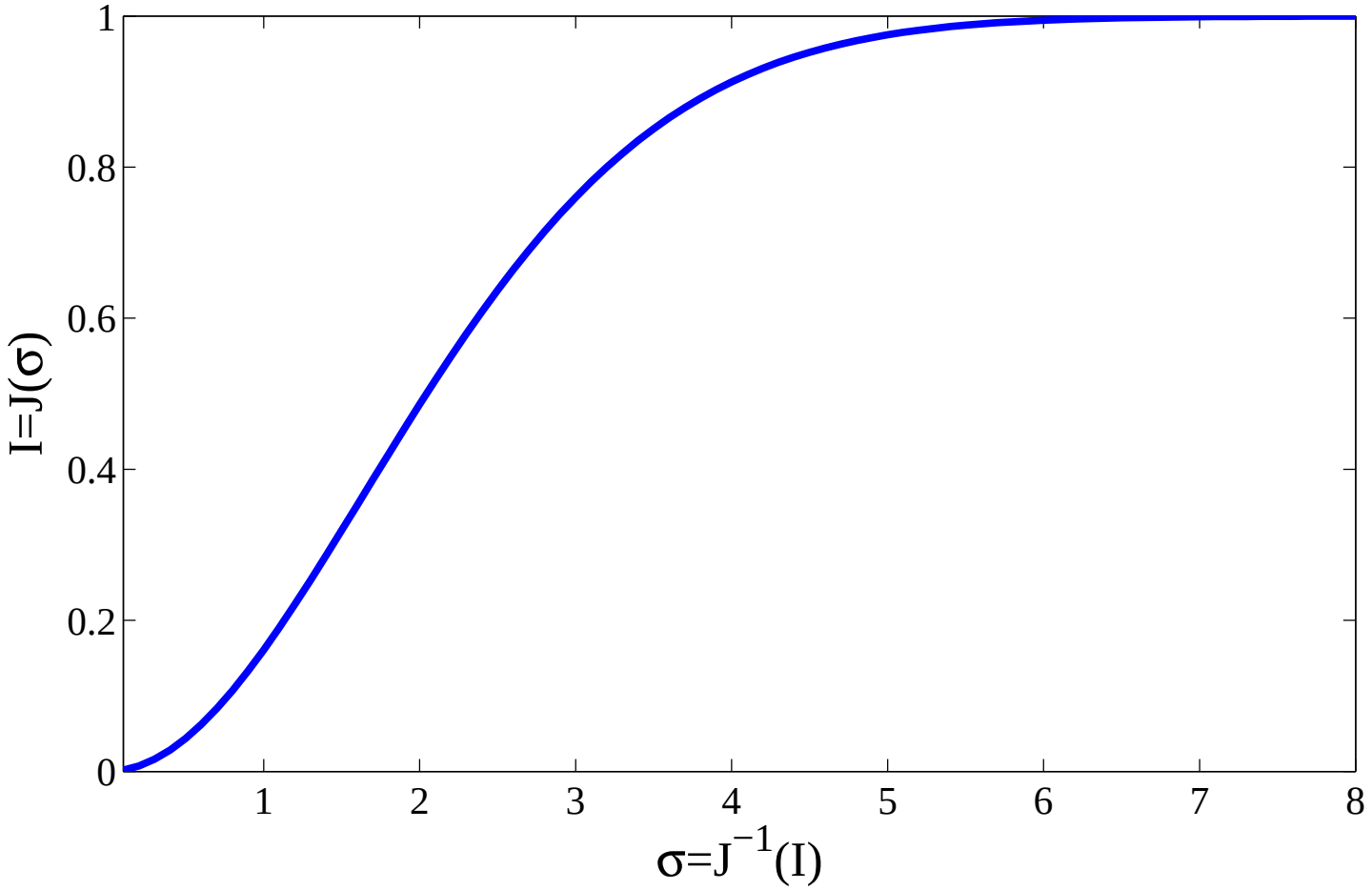
- $J(\sigma)$ **cannot be expressed in closed form. It is monotonically increasing in** $\sigma = 2/\widetilde{\sigma}_A$ **and thus invertible**

$$\sigma_A = J^{-1}(I_A)$$

- **Note also: The capacity of the binary input/continuous output AWGN channel is**

$$C = J(\sigma = 2/\widetilde{\sigma}_A)$$

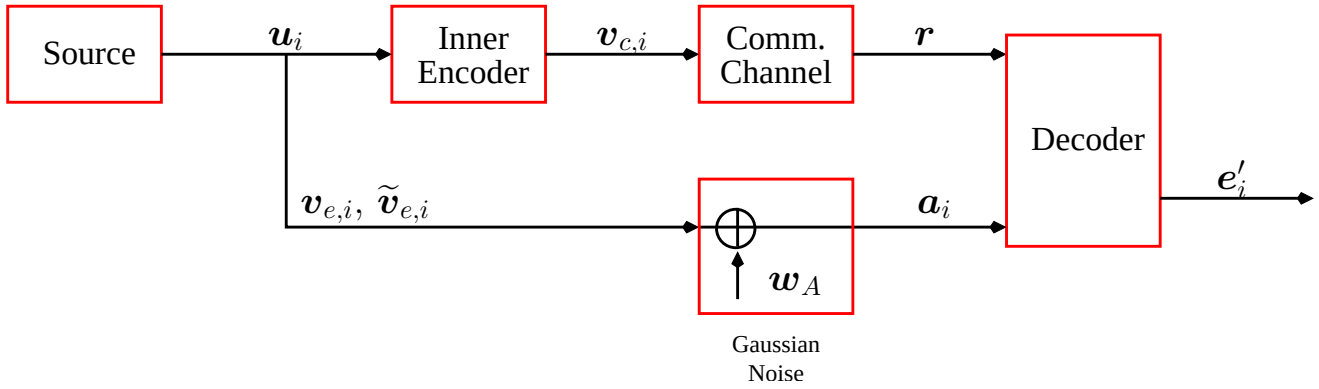
The $J(\cdot)$-function



EXIT For the Inner Decoder: Computing $I_{E_i'}$

- $I_{E_i'}$ **can be estimated by Monte-Carlo simulations (given** I_A **and** E_b/N_0 **) to get histogram approximations of** $p_{E'}(\xi|V_{e,i}=0)$, $p_{E'}(\xi|V_{e,i}=1)$ **.**
  Note: no Gaussian hypothesis is made for E'.
- **For a given** I_A, **the** $J(\sigma)$**-function provides the corresponding variance** $\sigma = (J^{-1}(I_A))^2$ **for simulations. The variance** $\widetilde{\sigma}_A^2 = 4/\sigma_A^2$ **is the variance before the log-likelihood computation.**

- **Simulation setup**



EXIT For the Inner Decoder: Computing $I_{E_i'}$ (cntd)

- **With the histogram estimate of $p_{E'}(\xi|V_{e,i} = v)$ we compute $I_{E'} = I(V_{e,i}; E')$ as**

$$I_{E'} = \frac{1}{2} \sum_{v_{e,i}=\{0,1\}} \int_{-\infty}^{+\infty} p_{E'}(\xi|V_{e,i} = v_{e,i}) \log_2 \frac{2p_{E'}(\xi|V_{e,i} = v_{e,i})}{p_{E'}(\xi|V_{e,i} = 0) + p_{E'}(\xi|V_{e,i} = 1)} \mathrm{d}\xi$$

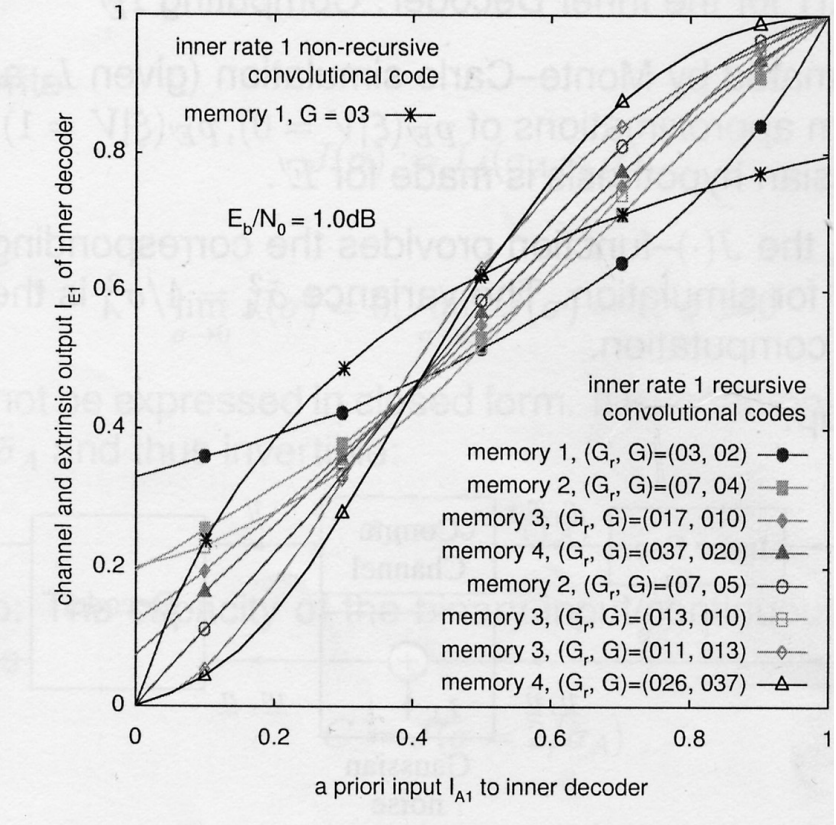
- **Viewing $I_{E'}$ as a function of I_A and E_b/N_0, the inner decoder EXIT function is written as**

$$I_{E'}(I_A, E_b/N_0)$$

or, for fixed E_b/N_0, just as

$$I_{E'}(I_A)$$

EXIT Functions: Inner Rate 1 Codes



Non-recursive codes do not go up to $I_{E'}(1) \approx 1$.

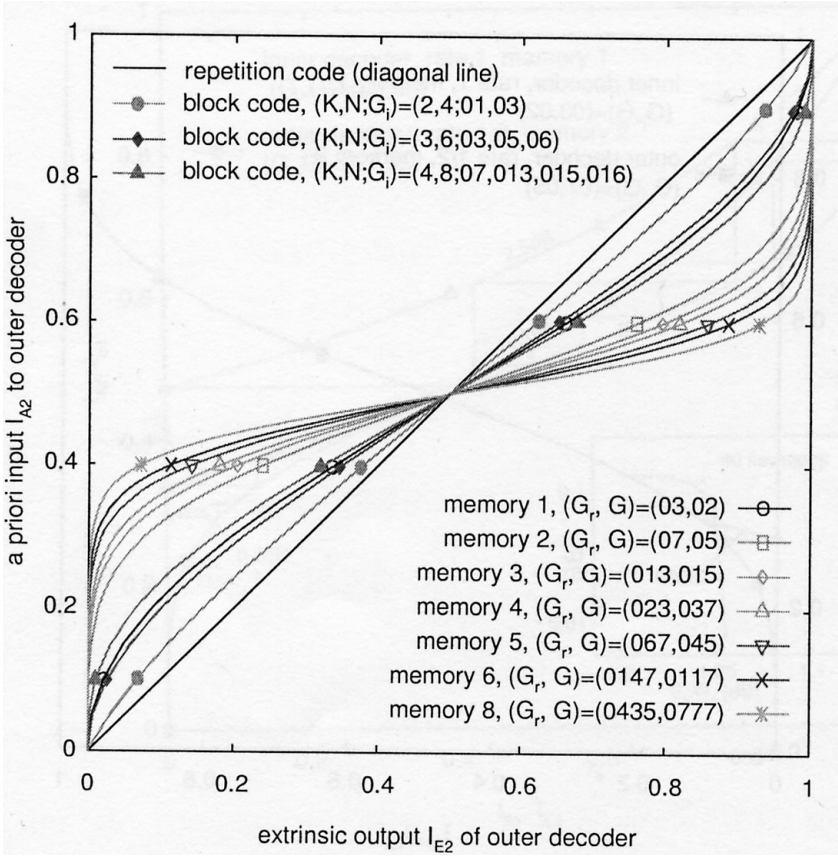
EXIT Functions for the Outer Decoder

- **The transfer function of the outer decoder is computed as above, but with $\boldsymbol{v}_{e,o} = \boldsymbol{v}_{c,o}$ and $\boldsymbol{r} = 0$.**
- **We compute $I_{A_o} = I(V_{c,o}; A_o)$, and the $I_{E_o} = I(V_{c,o}; E_o)$,to get the transfer function**

$$I_{E_o}(I_{A_o})$$

- **For the computation we again assume A_o to be Gaussian, and we obtain histogram approximations of $p_{E_o}(\xi|C_o = 0)$, $p_{E_o}(\xi|C_o = 1)$**
- **We plot $I_{E_o}(I_{A_0})$ with swapped axes: I_{A_0} is on the ordinate, I_{E_o} on the abscissa.**
- **This is in preparation of EXIT charts, where the inner and outer transfer functions are put on a single diagram.**

EXIT Functions: Outer Rate $1/2$ Codes



Rate $\frac{1}{2}$ repetition code has $I_{E_2} = I_{A_2}$.

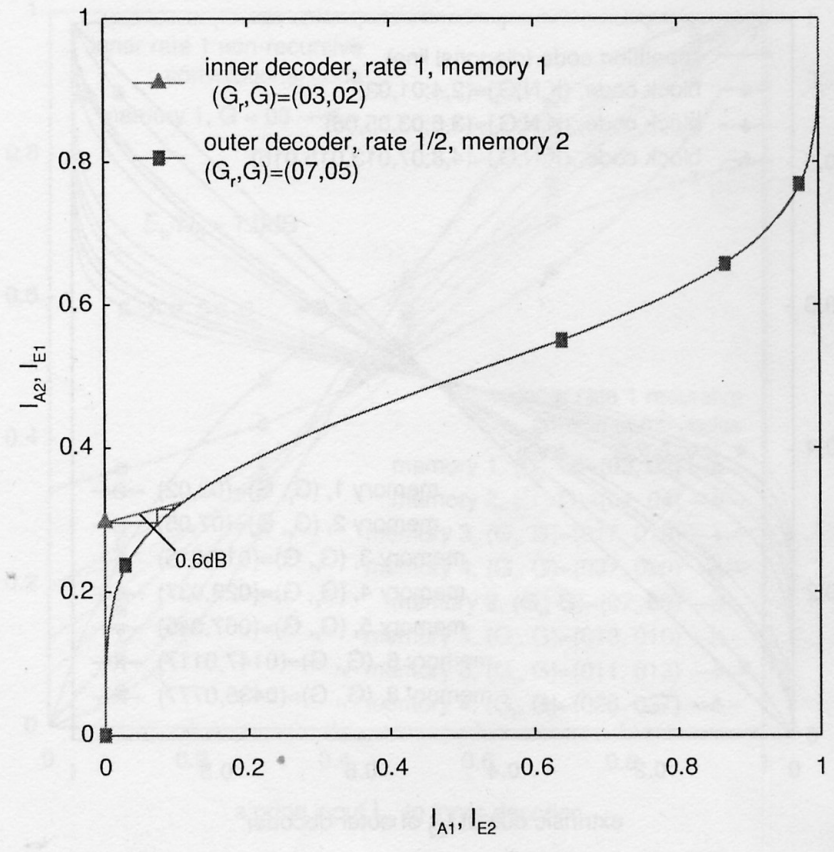
Extrinsic Information Transfer Charts

- To analyze the convergence of iterative decoding algorithms, both decoder EXIT functions (for inner and outer decoder) are plotted on a single diagram. For the second decoder axes are swapped

- This diagram is referred to as an extrinsic information transfer chart (EXIT chart), and the exchange of extrinsic information can be visualized as decoding trajectory.

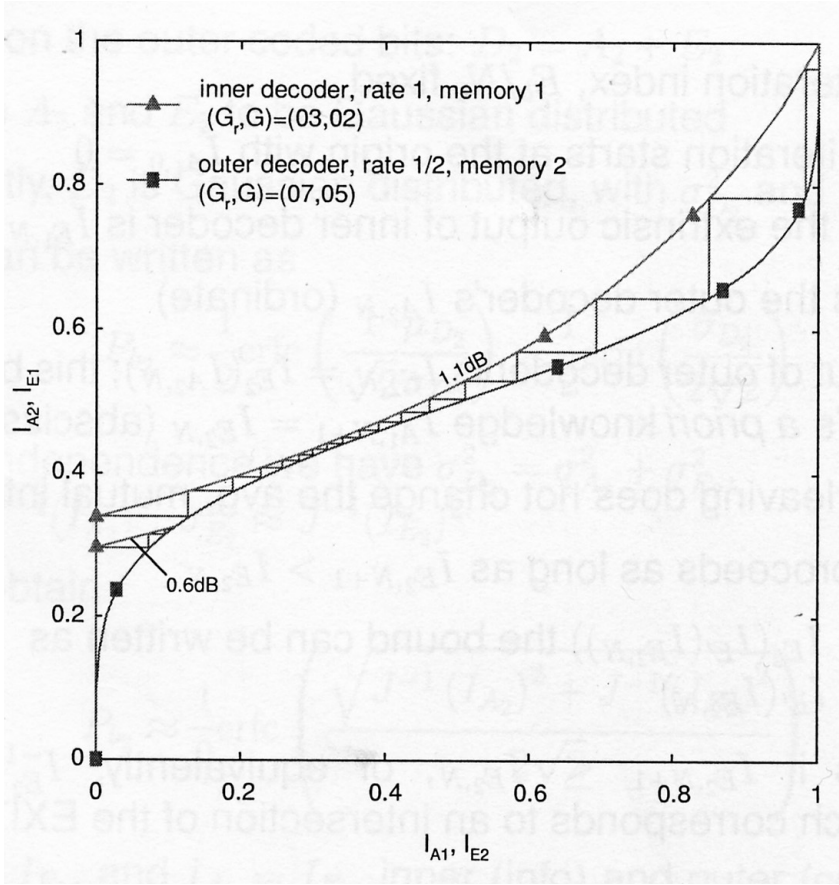
- Provided that the independence (memoryless, time-invariant) and Gaussian hypotheses are accurate for modelling extrinsic information (or a priori information), the decoding trajectory obtained by drawing a zigzag-path in the EXIT chart should match the trajectory computed by simulations.

EXIT Chart (1)



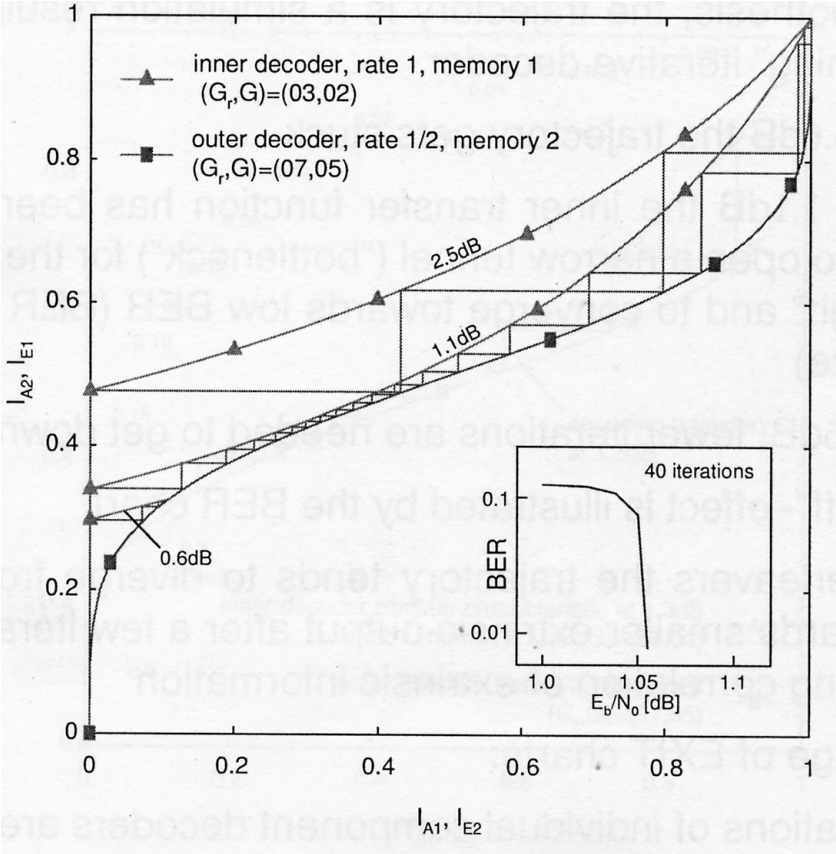
Interleaver size $4 \cdot 10^5$ coded bits; $\frac{E_b}{N_0} = 0.6$ dB: trajectory gets stuck.

EXIT Chart (2)



Interleaver size $4 \cdot 10^5$ coded bits; $\frac{E_b}{N_0} = 1.1$ dB: convergence to low BER through narrow tunnel.

EXIT Chart (3)



Interleaver size $4 \cdot 10^5$ coded bits; $\frac{E_b}{N_0} = 2.5$ dB: convergence tunnel wide open.

Remarks on EXIT Charts

- **Let N be the iteration index, E_b/N_0 fixed.**
- **For $N = 0$ the iteration starts at the origin with $I_{A_1,0} = 0$.**
- **At iteration N, the extrinsic output of inner decoder is $I_{E_1,N} = I_{E'}(I_{A_1,N})$.**
- **$I_{E_1,N}$ becomes the outer decoder's $I_{A_2,N}$ (ordinate).**
- **Extrinsic output of outer decoder is $I_{E_2,N} = I_{E_2}(I_{A_2,N})$; this becomes the inner decoder's a priori knowledge $I_{A_1,N+1} = I_{E_2,N}$ (abscissa).**
- **Note that the interleaving does not change the average mutual information.**
- **The iteration proceeds as long as $I_{E_2,N+1} > I_{E_2,N}$.**
- **With $I_{E_2,N+1} = I_{E_2}(I_{E_2'}(I_{E_2,N}))$ the bound can be written as $I_{E_2}^{-1}(I_{E_2,N+1}) > I_{E'}(I_{E_2,N})$.**
- **Iteration stops if $I_{E_2,N+1} \leq I_{E_2,N}$ or, equivalently, $I_{E_2}^{-1}(I_{E_2,N+1}) \leq I_{E'}(I_{E_2,N})$, which corresponds to an intersection of the EXIT functions.**

Remarks on EXIT Charts

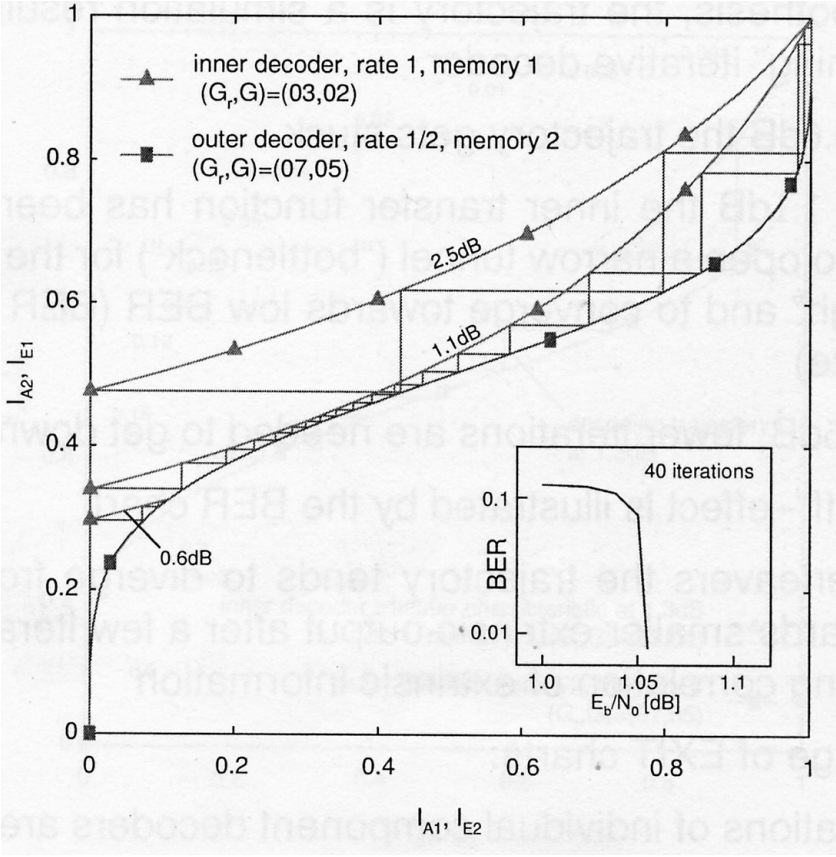
- **In the figures, the transfer functions are based on a Gaussian hypothesis, but the trajectory is a simulation result taken from the "free-running" iterative decoder.**
- **For $E_b/N_0 = 0.6$ dB the trajectory gets stuck.**
- **For $E_b/N_0 = 1.1$ dB the inner transfer function has been raised just high enough to open a narrow tunnel ("bottleneck") for the trajectory to "'sneak through" and to converge towards low BER (BER depends on interleaver size).**
- **At $E_b/N_0 = 2.5$ dB, fewer iterations are needed to get down to low BER.**
- **For short interleavers the trajectory tends to diverge from the EXIT functions towards smaller extrinsic output after few iterations, owing to an increasing correlation of extrinsic information.**
- **Main advantages of EXIT charts**
  1. **Only simulations of individual component decoders are required.**
  2. **Transfer function can be used in any combination.**

EXIT Chart for PC Codes: EXIT Functions

- The iterative decoder is symmetric in the sense that decoding for the second decoder with respect to A_2, E_2, Z_2 is essentially the same as for the first decoder with A_1, E_1, Z_1.
- We thus focus on the first decoder (index "1") omitted
- The input to the component decoder consists of channel observations Z and a priori knowledge A, with $Z = (2/\sigma^2)(x + w)$
- w is a realization of a Gaussian random variable having zero mean and variance $\sigma^2 = N_0/2$
- The appropriate decoder model is the same as for the inner code of a SC code because both a priori and channel log-likelihood ratios are available
- We again use the memoryless, time-invariant and Gaussian assumptions for the extrinsic channel.

EXIT Chart for a Parallel Concatenated Code

PC code, rate $\frac{1}{2}$, memory 4, interleaver 10^6, $(G_r, G) = (023, 037)$



References

- G. Kramer, **Turbo Codes and Iterative Decoding**, Mini Course at the Technische Universitat Wien, May 2003.
- S. Benedetto and E. Biglieri, **Principles of Digital Transmission with Wireless Applications**, Kluwer Academic, 1999.
- S. Lin and D. Costello, **Error Control Coding**, *Prentice Hall*, 1983.
- S. ten Brink, **Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes**, IEEE Trans. on Communications, vol. 49, n. 10, October 2001
- T. Richardson and R. Urbanke, **Modern Coding Theory**, *Cambridge University Press*, 2008.
- D. J. C. MacKay, **Information Theory, Inference, and Learning Algorithms**, *Cambridge University Press*, 2004.