Oct. 31, 2017 **EE 5161 Modern Coding Theory** Instructor: Pradeep Sarvepalli

Lecture # 09 **EXIT Charts of Concatenated Convolutional Codes** Scribe: Nived Rajaraman

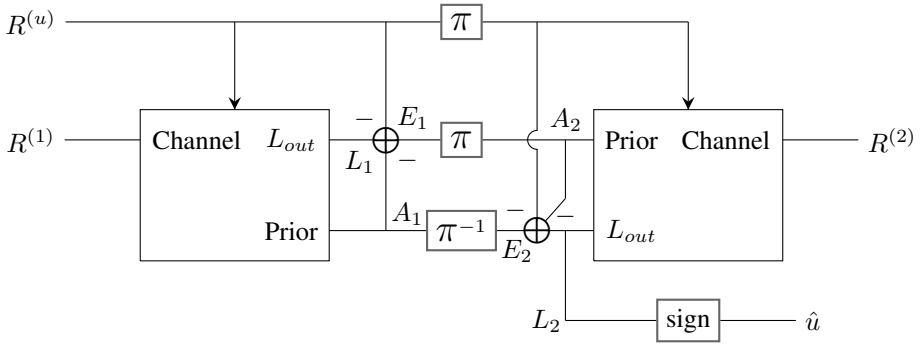
Summary: In this lecture, Serially Concatenated Convolutional Codes (SCCC) are analyzed from the lens of EXIT charts. We first begin with a recap of EXIT charts for Parallel Concatenated Convolutional Codes and subsequently develop the theory for Serially Concatenated Convolutional Codes. Finally we present Repeat Accumulate codes and discuss the relation between Turbo Codes and Factor Graphs.

References:

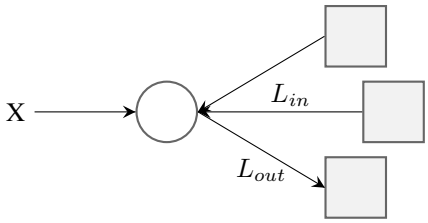
Chapters 5 and 6, *Iterative Error Correction*, Sarah Johnson

Introduction. Parallel Concatenated Convolutional Codes are a special type of iterative "Turbo" codes, which were first introduced in 1993 by Berrou, Glavieux and Thitimajshima. These codes perform extremely well in practice being high performance codes with a reduced complexity of encoding (via simple convolutional code circuits) and decoding. The decoding algorithm frequently used in practice is based on the Turbo Principle which involves the iterative exchange of information between the concatenated decoders. The Turbo principle can be used to generate a different class of *Serially Concatenated Convolutional Codes* that once again, can be decoded using an iterative external information transfer framework between the inner and outer decoders. We continue from the previous lecture and delve into an analysis of the performance of Parallel Concatenated Convolutional Codes using the framework of *EXIT charts*.

Parallel Concatenated Convolutional Codes Parallel Concatenated Convolutional Codes abbreviated PCCC are usually implemented as a parallel concatenation of two convolutional codes (usually both are chosen to be the same) followed by an optional puncturing scheme to improve the rate of the code. an iterative decoding algorithm for decoding such codes is detailed below. The two BCJR decoders used are used to decode the individual component codes of the overall code an information sharing approach is used to estimate the message as well as the redundant bits of both convolutional codes. $R^{(u)}$ denotes the LLR's of the message part of the received vector, while $R^{(1)}$ and $R^{(2)}$ correspond to the redundancy bits of the component codes. The block-wise interleaver is denoted by π. The decoder for the Turbo code is illustrated below:



EXIT Charts for Parallel Concatenated Convolutional Codes Given any decoding setup, we can represent the extrinsic information transfer for a node of the decoder as below. For the case of the PCCC turbo decoder, the nodes simply represent each of the BCJR decoders employed in the decoding process while L_{in} denotes the extrinsic information to a decoder (node) received from the other decoder (node). For LDPC codes, a similar setup is present - the individual codes being single parity check codes with the Tanner Graph serving to perform the information transfer between the different codes.



Generating EXIT Chart from Distributions To compute the EXIT chart of PCCC's we require $I(X;A_1)$ vs. $I(X,E_1)$ and $I(X;A_2)$ vs. $I(X,E_2)$. In order to do this, we would first have to generate the distributions of A_1, A_2, E_1 and E_2. We can neglect the role of the interleaver by assuming that E_1 and E_2 are identically distributed. Since there is no convenient closed form expression for E_1 or E_2, we numerically estimate their distributions in order to generate the EXIT chart.
We begin with an analysis of mutial information for an AWGN channel. The mutual information betewen the extrinsic LLR's and the codeword LLR's is illustrated below:

$$\begin{aligned} I(X;L) &= \int_E \sum_X P(X,E).log_2\left(\frac{P(X,E)}{P(X).P(E)}\right) \\ &= H(X) - H(X/L) \\ &= 1 - H(X/L) \\ &= 1 - \int f_L(l).log_2(1+e^{-1})dl = J(\sigma_L) \end{aligned}$$

Here $f_L(l)$ denotes the PDF of L and σ_L^2 denotes the variance of L. In our case, by estimating the PDF's of E_1 and E_2 as consistent Gaussian distributions, we follow the process listed below to compute the EXIT chart. We denote the mutual information $I(X;A_1)$ as I_{A_1} and $I(X;A_2)$ by I_{A_2}:

1. Start with a particular value of I_{A_1}.
2. From this, we estimate σ_L as $\sigma_L = J^{-1}(I_{A_1})$.
3. Using the identities:

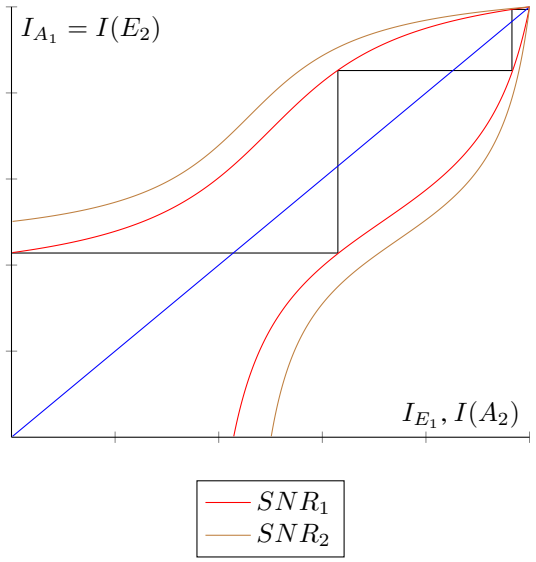
$$\begin{aligned} f(y/x=-1) &= f(y/x=1) \\ f(y/x=1) &= e^{-y}f(y/x=-1) \end{aligned}$$

   We generate the distribution of A_1 as $A_1 \sim N(\frac{\sigma_{A_1}^2}{2}, \sigma_{A_1})$.
4. We numerically compute $L_1 = BCJR(\text{trellis1}, R^{(u)}, R^{(1)}, A_1)$
5. Subsequently, we compute $E_1 = L_1 - R^{(u)} - A_1$

We repeat the above steps for many different received signals (generated based on a fixed value of $\frac{E_b}{N_0}$). The distribution for E_1 is generated by sampling. Thus we generate histogram for the PDF of E_1 for a given value of I_{A_1}. The final step is to estimate $I(X;E_1)$ which is once again motivated based on the analysis performed above:

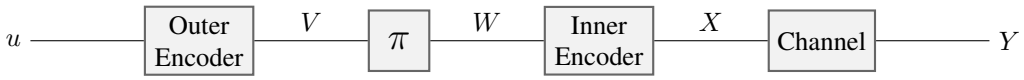
$$I(X;E_1) = 1 - E\left[log_2\left(1+e^{-E_1}\right)\right]$$

In order to generate the EXIT chart, we repeat the above process for many different values of I_{A_1}. The plot of I_{E_1} vs I_{A_1} is the EXIT chart for the 1^{st} decoder. A similar process is repeated to generate the EXIT chart of the 2^{nd} decoder. The diagram below displays the EXIT chart generated for 2 different signal SNR's ($\frac{E_b}{N_0}$ in dB):



As the SNR of the received signal degrades, the two curves (represented with the same color) approach closer to each other ($SNR_1 < SNR_2$). When the two graphs touch each other, i.e. when the tunnel closes, the corresponding SNR of the received signal gives the noise threshold of the decoder.
With this analysis in place, we now move onto Serially Concatenated Convolutional Codes.

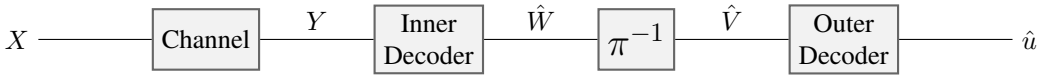
Serially Concatenated Convolutional Codes Serially Concatenated Convolutional Codes consist of a setup with 2 serially connected component codes. The corresponding decoder is also composed of 2 component decoders that use external information transfer to compute the estimate for the transmitted codeword. The arrangement is motivated based on the Turbo principle as for in Parallel Concatenated codes. The setup consists of an inner and an outer encoder/decoder that work in combination and as before, an interleaver is used in conjunction with the two to reduce the proportion of bad inputs to the encoder.



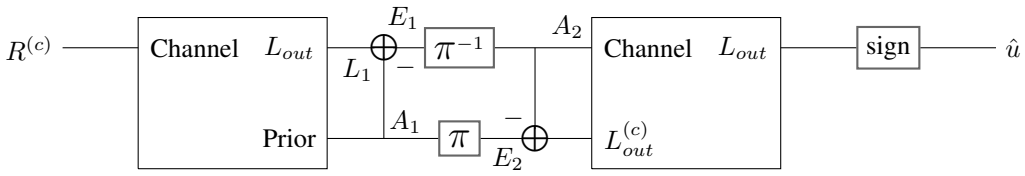
If the interleaver is placed either before the outer encoder or after the inner encoder, it has no effect as it simply permutes the bad inputs (which does not reduce the set of low weight parity sequences) or permutes the corresponding low weight parity sequences, which once again does not provide any performance gain.

If the outer encoder is a recursive encoder (RE), but the inner encoder is not a recursive encoder (NRE), then a bad input would still generate an overall bad output. However, if the inner encoder is recursive, then the proportion of bad outputs would be far fewer (provided the interleaver does not perform a trivial shift operation). Thus, the outer encoder can be a recursive or non-recursive encoder, but the inner encoder should be a recursive encoder.

Similarly the decoder can be illustrated by a block diagram as below:

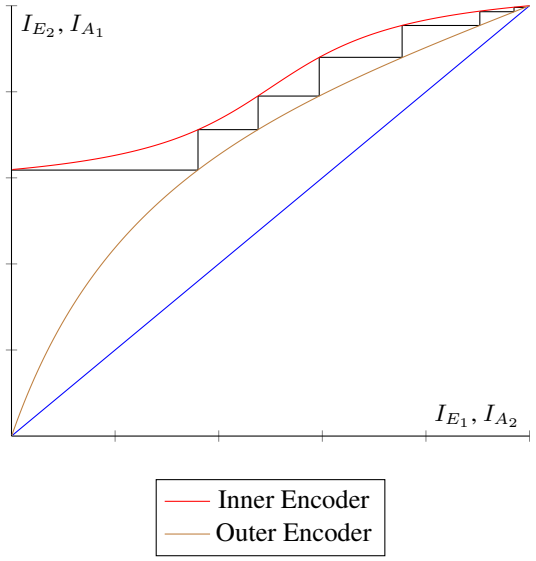


And can be designed as below:



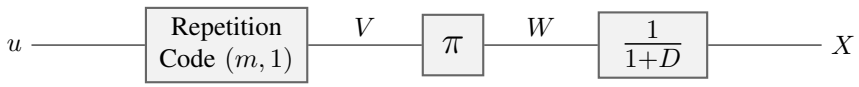
The first decoder is termed the inner decoder (as it decodes the inner convolutional code, similarly the second decoder is the outer decoder. The interconnection between the two is symbolic of the external information transfer between the component codes. The first BCJR decoder generates the LLR's that estimate W, and pass on this after de-interleaving as a prior for V to the outer decoder. The inner decoder and interleaver together appear to be like the channel to the outer decoder, and it generates an estimate for V and passes on this (after interleaving) as a prior to the inner decoder.

EXIT charts for SCCC codes The EXIT chart for SCCC codes is quite similar to that of PCCC codes, in that the inner decoder behaves similar to that of the parallel concatenated encoders. The exception here is that the outer decoder does not have any information about the codeword from the channel, and therefore, when I_{A_2} is 0, I_{E_2} must be 0. Put together, the EXIT chart looks as below:



It is evident that SCCC codes would have higher rate than PCCC codes, but this comes at the cost of reduced SNR threshold for reliable decoding. From the diagram below it can be seen that the tunnel is smaller for SCCC and PCCC codes at the same SNR and that the threshold would be reached first for SCCC codes on decreasing the signal SNR.

Repeat Accumulate (RA) Codes Repeat Accumulate Codes are closely related to SCCC codes. The name comes from the 2 step encoder, the first being a $(m, 1)$ repetition code and the second being an accumulator (convolutional code). The block diagram for RA code encoders is as below:



Turbo codes and Factor Graphs The bitwise decoding problem can be represented as:

$$u_t^{(j)} = \underset{u_t^{(j)}}{\text{argmax}}\, P(u_t^{(j)}/Y) = \underset{u}{\text{argmax}} \sum_{\sim u_t^{(i)}} P(u/Y)$$

For BCJR decoders, this turns out to be of the form:

$$u_t^{(j)} = \underset{u}{argmax} \sum_{\sim u_t^{(j)}} \sum_{(S_r, S_s)} P(S_r, S_s, Y)/P(Y)$$

Thus, there is a close link between factor graphs and Turbo decoders. This can be established as below:

1. For the first component code, the received LLR's from the channel and the extrinsic LLR's (priors) from the other half of the decoder are passed onto the bit-nodes. the channel are fed into the bit nodes. Values of Γ are then generated using the messages from the bit nodes.

2. For the trellis of the first code, α messages are passed from left to right through the state nodes, β messages are passed from right to left through the state nodes, and extrinsic LLRs are calculated and passed to the systematic-bit nodes.

3. Repeat (1) and (2) for the second component code.

Note that unlike in an LDPC code where the factor graph can be computed in parallel, the same is not possible in a Turbo code as the α and β messages must be passed one at a time through the factor graph.