Code Design with EXIT Charts

Ingmar Land

October 2, 2013
1 Introduction

Iterative decoding provides high performance at low complexity for many state-of-the-art codes, like turbo codes [1], low-density parity-check (LDPC) codes [2], and irregular repeat accumulate codes [3, 4], as well as for many iterative receiver structures. In order to achieve such high performance, the codes need to be designed such that iterative decoding works well. The extrinsic information transfer (EXIT) analysis [5, 6] is a powerful tool to analyse the iterative decoding process and to accomplish the required code design. Furthermore it provides intuition for the decoding process, and also allows to formulate the design problem as a convex optimisation problem.

EXIT charts were introduced by ten Brink for analysis and design of coded modulation [7], parallel concatenated codes (parallel turbo codes) [5], and serially concatenated codes (serial turbo codes) [6]. EXIT charts were also proposed for efficient design of coded modulation with low-density parity-check (LDPC) codes [8] and irregular repeat accumulate (IRA) codes [9]. Further analysis of EXIT functions lead to various important theorems, like the area theorem and the duality theorem [10]. Bounds of EXIT functions for application to LDPC codes were found by information combining [11–14]. The EXIT analysis is now a well developed engineering tool and can be found in various overview papers and textbooks, e.g. [15–22].

This chapter reviews the EXIT chart method from a practical point of view and shows how to apply it to the design of parallel concatenated codes, serially concatenated codes and LDPC codes. We restrict ourselves to the basic principles and concepts of the EXIT analysis. For this purpose, we consider the transmission of binary linear channel codes over memoryless symmetric channels and iterative decoders. Examples are provided to illustrate the ideas. At the end of this chapter, we conclude with some remarks and comments and outline further properties, generalisations and applications. In our descriptions we assume that the reader is familiar with concatenated codes, LDPC codes and iterative decoding.

System model

The system model is depicted in Fig. 1. Assume a binary linear code of length $N$, dimension $K$ and rate $R = K/N$. A binary symmetric source (BSS) generates an information word $u$ of length $K$. The encoder maps this to the codeword $x$ of length $N$. This codeword is transmitted over the memoryless symmetric communication channel defined by the transition probabilities $p_{Y|X}(y|x)$ [23]. Examples for such channels are the binary symmetric channel (BSC), the binary erasure channel (BEC) and the AWGN channel with BPSK mapping (BI-AWGNC).

The channel output vector (observation vector) $y$ is given to the decoder, in which two component decoders operate in an iterative fashion. At the end of this iteration, the decoder computes the information word estimate $\hat{u}$. The component decoders are

\[^{1}\]A binary-input channel is called symmetric if for any output value $y$ there is an output value $y'$ such that $p_{Y|X}(y|0) = p_{Y|X}(y'|1)$ [13, 23].
assumed to be a-posteriori probability (APP) decoders (see discussion in Section 5).

**Notation**

For a vector \( \mathbf{a} = [a_1 a_2 \ldots a_L] \), the permuted (interleaved) vector according to a permutation vector \( \pi \) is denoted by \( \text{perm}_\pi \mathbf{a} = [a_{\pi_1} a_{\pi_2} \ldots a_{\pi_L}] \); the vector with element \( a_l \) excluded is denoted by \( \mathbf{a} \setminus l = [a_1 \ldots a_{l-1} a_{l+1} \ldots a_L] \).

Random variables are denoted by uppercase letters and realisations by the corresponding lowercase letters. For two random variables \( X \) and \( Y \), entropy, conditional entropy and mutual information are denoted by \( H(X) \), \( H(X|Y) \), and \( I(X;Y) \), respectively. Further, \( H_2(p) = -p \log_2 p - (1-p) \log_2 (1-p) \) denotes the binary entropy function [24].

**2 Parallel concatenated codes**

This section deals with the analysis and design of parallel concatenated codes, like the famous Turbo codes invented by Berrou and Glavieux [1, 25]. After reviewing the encoder and the decoder, we introduce the decoding model for the component decoders and the EXIT analysis. Then we show how to use this method for code design.

**2.1 Encoder and decoder**

Consider a parallel concatenated (PC) code. As depicted in Fig. 2, the PC encoder is defined by two component encoders and an interleaver \( \pi \). The input to Encoder 1 is the original information word, \( \mathbf{u}_1 = \mathbf{u} \), of length \( K \), and the output is the codeword \( \mathbf{x}_1 \) of length \( N_1 \). The input to Encoder 2 is the interleaved information word, \( \mathbf{u}_2 = \text{perm}_\pi \mathbf{u}_1 \), of length \( K \), and the output is the codeword \( \mathbf{x}_2 \) of length \( N_2 \). Thus the PC encoder maps the information word \( \mathbf{u} \) of length \( K \) to the PC codeword \( \mathbf{x} = [\mathbf{x}_1 \mathbf{x}_2] \) of length \( N = N_1 + N_2 \).

A typical example for a PC code is a Turbo code of rate 1/3, where Encoder 1 is a recursive systematic convolutional encoder of rate 1/2, and Encoder 2 is a recursive convolutional encoder of rate 1.

The codeword \( \mathbf{x} \) is transmitted over the communication channel, yielding the observation vector \( \mathbf{y} \). The observation vector is split up as \( \mathbf{y} = [\mathbf{y}_1 \mathbf{y}_2] \), where \( \mathbf{y}_1 \) corresponds
Figure 2: PC encoder.

to $x_1$, and $y_2$ corresponds to $x_2$.

Figure 3: PC decoder.

The iterative decoder for the PC code\(^2\) is shown in Fig. 3. The observation vector $y_1$ is fed to Decoder 1, and the observation vector $y_2$ is fed to Decoder 2. We assume that both decoders are a-posteriori probability (APP) decoders producing extrinsic probabilities\(^3\). Decoder 1 computes the vector of extrinsic values $e_1$ based on the observation $y_1$ and the a-priori values $a_1$ coming from Decoder 2. At iteration $l$, $l = 1, 2, \ldots, L$, we have

$$e_{1k}^{(l)} := P(U_{1k} = 0|y_1, a_1^{(l)}), \quad (1)$$

$k = 1, 2, \ldots, K$. As in iteration 1, no a-priori values are available, we define $a_1^{(1)} = [1/2 \ldots 1/2]$. Notice that $a_{1k}^{(l)}$ is not used to compute $e_{1k}^{(l)}$, and thus $e_{1k}^{(l)}$ is an extrinsic\(^4\)

\(^2\)Some authors describe the turbo decoder in a “symmetric” form, where the channel observations for the systematic bits (information bits) are fed to both component decoders. In this case, the extrinsic values need to be defined in a slightly different way.

\(^3\)Equivalently, extrinsic L-values may be exchanged between the component decoders. Note however, that APP decoders are required as otherwise the EXIT analysis is not correct. The often used MaxLogAPP decoder [26], for example, can not be properly analysed with the EXIT chart method, see Section 5.

\(^4\)The extrinsic probability of a bit is a special a-posteriori probability of this bit, where the a-priori probability of this bit is excluded from the condition.
probability. Interleaving $e_1^{(l)}$ gives the a-priori values

$$a_2^{(l)} = \text{perm}_x e_1^{(l)}$$

(2)

for Decoder 2. Then Decoder 2 computes the vector of extrinsic values $e_2$ based on the observation $y_2$ and the a-priori values $a_2$ coming from Decoder 1. At iteration $l$, we have

$$e_2^{(l)} := P(U_{2k} = 0 | y_2, a_2^{(l)})$$

(3)

$k = 1, 2, \ldots, K$. De-interleaving $e_2^{(l)}$ gives the a-priori values

$$a_1^{(l+1)} = \text{perm}_{x-1} e_2^{(l)}$$

(4)

for Decoder 1. After $L$ iterations (or after a certain stopping criterion is fulfilled), Decoder 1 (or similarly Decoder 2) computes the estimate $\hat{u}$ of the information word:

$$\hat{u}_k = \begin{cases} 
0 & \text{if } P(U_k = 0 | y_1, a_1^{(L)}) \geq 1/2, \\
1 & \text{otherwise,} 
\end{cases}$$

(5)

$k = 1, 2, \ldots, K$. Notice that all probabilities are typically efficiently computed in the code trellis using some kind of forward-backward algorithm [26–28].

### 2.2 The EXIT method

Assume now decoding of a specific observation $y$ and the a-priori and extrinsic vectors resulting from the decoder activations, as listed in Table 1 in the first three columns. Remember that Decoder 1 does not have any a-priori information in the first iteration, and therefore all entries of $a_1^{(1)}$ are set to $1/2$. (The permutations are omitted in the table.) The final decision of the information bit estimates is omitted, as we focus here on the evolution during the iterations.

<table>
<thead>
<tr>
<th>Iteration number</th>
<th>Decoder</th>
<th>Decoder operation</th>
<th>Information transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l = 1$</td>
<td>Decoder 1</td>
<td>$a_1^{(1)} \rightarrow e_1^{(1)}$</td>
<td>$I_{A1}(a_1^{(1)}) \rightarrow I_{E1}(e_1^{(1)})$</td>
</tr>
<tr>
<td></td>
<td>Decoder 2</td>
<td>$a_2^{(1)} \rightarrow e_2^{(1)}$</td>
<td>$I_{A2}(a_2^{(1)}) \rightarrow I_{E2}(e_2^{(1)})$</td>
</tr>
<tr>
<td>$l = 2$</td>
<td>Decoder 1</td>
<td>$a_1^{(2)} \rightarrow e_1^{(2)}$</td>
<td>$I_{A1}(a_1^{(2)}) \rightarrow I_{E1}(e_1^{(2)})$</td>
</tr>
<tr>
<td></td>
<td>Decoder 2</td>
<td>$a_2^{(2)} \rightarrow e_2^{(2)}$</td>
<td>$I_{A2}(a_2^{(2)}) \rightarrow I_{E2}(e_2^{(2)})$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$l = L$</td>
<td>Decoder 1</td>
<td>$a_1^{(L)} \rightarrow e_1^{(L)}$</td>
<td>$I_{A1}(a_1^{(L)}) \rightarrow I_{E1}(e_1^{(L)})$</td>
</tr>
<tr>
<td></td>
<td>Decoder 2</td>
<td>$a_2^{(L)} \rightarrow e_2^{(L)}$</td>
<td>$I_{A2}(a_2^{(L)}) \rightarrow I_{E2}(e_2^{(L)})$</td>
</tr>
</tbody>
</table>

Table 1: Decoder operations (columns 1-3) and associated information transfer (column 4).
2.2.1 Decoding trajectory

Each of the vectors of a-priori and extrinsic values contains information about the transmitted bits. (Note that the permutations do not change the information content.) If the overall decoding is successful, the amount of information will increase with each activation of Decoder 1 and Decoder 2. To measure this “information”, we use the average bit-wise mutual information about information bits.

This information before decoding for a given vector \( \mathbf{a}_1 \) is the bit-wise mutual information between information bits and these a-priori values, called the a-priori information:

\[
I_{A1}(\mathbf{a}_1) := \frac{1}{K} \sum_{k=1}^{K} I(U_{1k}; A_{1k} = a_{1k}).
\]  

(6)

The mutual information can be written as

\[
I(U_{1k}; A_{1k} = a_{1k}) = H(U_{1k}) - H(U_{1k}|A_{1k} = a_{1k}) = 1 - H_2(a_{1k}),
\]

where \( H_2(.) \) denotes the binary entropy function; we use that \( P(U_{1k} = 0|A_{1k} = a_{1k}) = a_{1k} \) by definition of \( a_{1k} \). Thus we can rewrite the a-priori information as

\[
I_{A1}(\mathbf{a}_1) = 1 - \frac{1}{K} \sum_{k=1}^{K} H_2(a_{1k}).
\]  

(7)

Similarly, the information after decoding is the bit-wise mutual information between information bits and extrinsic values, called the extrinsic information:

\[
I_{E1}(\mathbf{e}_1) := \frac{1}{K} \sum_{k=1}^{K} I(U_{1k}; E_{1k} = e_{1k}).
\]

(8)

Like above, we may rewrite this as

\[
I_{E1}(\mathbf{e}_1) = 1 - \frac{1}{K} \sum_{k=1}^{K} H_2(e_{1k}).
\]  

(9)

where we exploit again that \( P(U_{1k} = 0|E_{1k} = e_{1k}) = e_{1k} \) by definition of \( e_{1k} \). For Decoder 2, we similarly define the a-priori information as

\[
I_{A2}(\mathbf{a}_2) := \frac{1}{K} \sum_{k=1}^{K} I(U_{2k}; A_{2k} = a_{2k})
\]  

(10)

and the extrinsic information as

\[
I_{E2}(\mathbf{e}_2) := \frac{1}{K} \sum_{k=1}^{K} I(U_{2k}; E_{2k} = e_{2k}).
\]  

(11)
which may be computed as above.

Using this measure of information, we can associate a-priori information with every vector of a-priori values and extrinsic information with every vector of extrinsic values. Every decoding operation leads thus to an information transfer from a-priori information to extrinsic information. This is shown in Table 1 in column 4. It is obvious that the vectors $e_l^{(1)}$ and $a_l^{(1)}$ contain the same information, i.e., $I_{E1}(e_l^{(1)}) = I_{A2}(a_l^{(1)})$; similarly $e_2^{(l)}$ and $a_1^{(l+1)}$ contain the same information, i.e., $I_{E2}(e_2^{(l)}) = I_{A1}(a_1^{(l+1)})$. As every component decoder is characterised by this transfer of extrinsic information, the method is called the \textit{extrinsic information transfer (EXIT) analysis}.

![EXIT chart diagram](image)

\textbf{Figure 4: Example of a decoding trajectory in the EXIT chart.}

Fig. 4 depicts the corresponding decoding trajectory. Starting from the origin with $I_{A1}(a_1^{(1)}) = 0$, Decoder 1 produces $I_{E1}(e_1^{(1)})$, depicted by the line going upwards. Given $I_{A2}(a_2^{(1)}) = I_{E1}(e_1^{(1)})$, Decoder 2 produces $I_{E2}(e_2^{(1)})$, depicted by the line going to the right. This continues throughout the iteration. If the trajectory reaches the top border of the EXIT chart, we have $I_{E1}(e_1^{(L)}) = 1$, and if it reaches the right border, we have $I_{E2}(e_2^{(L)}) = 1$. In either case, the information is maximal and thus error-free decoding is guaranteed.

We wish now to use this concept of extrinsic information transfer for analysis and design. For a given code and communication channel, the trajectory depends on the specific realisation of the observation vector $y$. To eliminate this effect, we consider the asymptotic case of infinite length codes. The observation vector then becomes typical (in the information-theoretic sense [24]) and the trajectory becomes deterministic.

© Ingmar Land, October 2, 2013 7
Instead of looking at the a-priori information and the extrinsic information for a specific given a-priori vector or extrinsic vector, we then define for Decoder 1 the average a-priori information as

$$I_{A1} := \frac{1}{K} \sum_{k=1}^{K} I(U_{1k}; A_{1k}),$$

and the average extrinsic information as

$$I_{E1} := \frac{1}{K} \sum_{k=1}^{K} I(U_{1k}; E_{1k});$$

Similarly, we define for Decoder

$$I_{A2} := \frac{1}{K} \sum_{k=1}^{K} I(U_{2k}; A_{2k}),$$

and

$$I_{E2} := \frac{1}{K} \sum_{k=1}^{K} I(U_{2k}; E_{2k});$$

As before, we have

$$I_{A2} = I_{E1}$$

and

$$I_{A1} = I_{E2}.$$

In above expressions we have dropped the super-indices \(l\) for the iteration numbers, to simplify the notation; the values are clear from the context.

Note that we have \(I_{A1}(a_1) \to I_{A1}\) for \(N \to \infty\) with probability 1 (by typicality), and this holds similarly for \(I_{A2}, I_{E1}\) and \(I_{E2}\). For convenience we may refer to the average a-priori and extrinsic information simply as a-priori and extrinsic information, respectively.

So far the whole approach has only been used to visualise the decoding process over the iteration, in particular to visualise the evolution of the information about the transmitted bits. For this purpose, it is required to run the iterative decoder and track the decoding process. As this is a very time-consuming process, it is desirable to be able to predict the trajectory without actually running the iterative decoder. Such a method is now developed.

### 2.2.2 Decoding model and transfer function

Consider first Decoder 1. (For convenience we omit the index \(l\) for the iteration numbers.) In every iteration step, the decoder obtains observations for its code bits, provided by the communication channel, and a-priori values for its information bits, provided by Decoder 2. Therefore for Decoder 1, we have the decoding model depicted in Fig. 5.
Notice that we model the statistical connection between the information word $\mathbf{u}_1$ and the vector $\mathbf{a}_1$ of a-priori values as a virtual channel, termed a-priori channel\(^5\). In general, the elements of $\mathbf{a}_1$ may be correlated, and thus the a-priori channel may not be memoryless. Assuming very long codes and good interleavers, however, we may ignore this correlation and assume the a-priori channel as memoryless. Furthermore, if the communication channel is symmetric and the code is linear, the a-priori channel is a symmetric channel as well.

The channel model for the a-priori channel is to be chosen such that it approximates well the actual transition probabilities observed in the iterative decoder. If the communication channel is a BEC, the a-priori channel is also a BEC, as can be shown. For other models of the communication channel, the BI-AWGNC usually provides a good model to approximate the true a-priori channel (in the iterative decoder). In general any channel model may be applied, that has only one parameter (like the erasure probability for the BEC or the signal-to-noise ratio (SNR) for the BI-AWGNC) and is a degraded\(^6\) channel.

For a given communication channel and given a-priori channel, we define now the EXIT function of Decoder 1 as

$$T_1 : [0, 1] \to [0, 1]$$

$$I_{A1} \leftrightarrow I_{E1} = T_1(I_{A1}).$$

(18)

Similarly the EXIT function of Decoder 2 is defined as

$$T_2 : [0, 1] \to [0, 1]$$

$$I_{A2} \leftrightarrow I_{E2} = T_2(I_{A2}).$$

(19)

Using the decoding model, these EXIT functions can easily be determined by simulation. The communication channel is kept fixed. The parameter of the a-priori channel is changed such that the a-priori information varies between 0 and 1; for every value of a-priori information, the extrinsic information is determined.

An example for the EXIT functions of a PC code is depicted in Fig. 6. The EXIT function $T_1$ of Decoder 1 maps the a-priori information $I_{A1}$ (x-axis) to the extrinsic information $I_{E1}$ (y-axis). Similarly, the EXIT function $T_2$ of Decoder 2 maps the a-priori information $I_{A2}$ (x-axis) to the extrinsic information $I_{E2}$ (y-axis).

\(^5\)This channel is also called virtual extrinsic channel in the literature.

\(^6\)Here degradation means that the mutual information of the channel is strictly monotonic in the channel parameter.
Figure 6: Example of EXIT functions of PC code, including the decoding trajectory, where error-free decoding is possible.

information $I_{A2}$ (y-axis) to the extrinsic information $I_{E2}$ (x-axis). Note that the roles of the x-axis and the y-axis are swapped for $T_2$, to reflect (16) and (17).

The EXIT functions allow to easily predict the decoding trajectory by reflecting the values on the corresponding EXIT functions. (We now use superindices again to indicate the iteration number.) Starting at the origin with $I_{A1}^{(1)} = 0$, we obtain the following sequence of information values for iteration $l = 1$:

\[
I_{E1}^{(1)} = T_1(I_{A1}^{(1)}), \quad I_{A2}^{(1)} = I_{E1}^{(1)} \quad I_{A1}^{(1)} = I_{E2}^{(1)}
\]

and for iteration $l = 2$:

\[
I_{E1}^{(2)} = T_1(I_{A1}^{(2)}), \quad I_{A2}^{(2)} = I_{E1}^{(2)} \quad I_{A1}^{(2)} = I_{E2}^{(2)}.
\]

Continuing similarly until iteration $l = L$ gives the complete trajectory.

If the code is very long, the decoding trajectory predicted from the EXIT functions and the real trajectory measured during the iterative decoding process match very well [5]. Note that any difference between the predicted trajectory and the measured (true) trajectory is only due to the difference between the model for the a-priori channel used to determine the EXIT functions and the true a-priori channel occurring during iterative
decoding. As noted earlier, if the communication channel is a BEC, then using a BEC for the a-priori channel is exact.

If the decoding trajectory reaches $I_{E1} = 1$ (or $I_{E2} = 1$), then Decoder 1 (or Decoder 2) has full information about the transmitted bits and can estimate the information bits without errors. Fig. 6 shows an example where error-free decoding can be achieved with about $t = 4$ iterations. Fig. 7 depicts an example, where error-free decoding is impossible, independent of how many iterations are used: the two EXIT functions cross and correspondingly the decoding trajectory gets stuck, and thus error-free decoding is impossible. In fact, we can predict that after 4 to 5 iterations, the decoding outcome will not significantly change any more.

![EXIT functions](image)

Figure 7: Example of EXIT functions for a PC code, including the decoding trajectory, where error-free decoding is not possible.

For given models of the communication channel and a-priori channel, the shape of the EXIT function depends on the code, the encoder, and the parameter of the communication channel\(^7\) (e.g. the SNR for the BI-AWGNC or the erasure probability for the BEC). As explained above, inspecting the EXIT functions allows to predict whether iterative decoding results in error-free decoding or not. Thus this method can be used to analyse and design codes. The examples so far are only for illustration of the concepts and ideas of the EXIT chart method. The next section discusses the application to real codes.

\(^7\)The chosen decoding algorithm may also affect the shape of the EXIT function; however, for non-APP component decoders, the EXIT analysis is not reliable, as discussed in Section 5.
2.3 Code analysis and design

As an example consider a parallel concatenated convolutional code (turbo code) of rate 1/3. (Remember that the code is assumed to have infinite length.) Encoder 1 is a recursive systematic convolutional encoder of rate 1/2, defined by the generator \( g_1(D) \); Encoder 2 is a recursive convolutional encoder of rate 1, defined by the generator \( g_2(D) \), where

\[
g_1(D) = \left( 1, \frac{f_1(D)}{r_1(D)} \right) \quad \quad g_2(D) = \frac{f_2(D)}{r_2(D)}.
\]

The polynomials are specified in the common octal notation, like in Table 2. The interleaver is randomly chosen. Assume that the communication channel is a BI-AWGNC with an SNR of \( E_s/N_0 \). Assume further a BI-AWGNC to model the a-priori channel.

![Figure 8: EXIT functions of a turbo code for \( E_s/N_0 \) between \(-7 \) dB and \(-4 \) dB in steps of \(0.25 \) dB. (EXIT functions are depicted up to their intersection.) The decoding threshold is at \( E_s/N_0 \approx -4.75 \) dB, where the two EXIT functions just touch each other.](image)

Fig. 8 shows the EXIT functions for \( g_1 = (1, 5/7) \) and \( g_2 = (5/7) \) for various SNR values of the communication channel. The EXIT function \( T_1 \) of Decoder 1 starts with a non-zero extrinsic information, \( I_{E1} > 0 \), for zero a-priori information, \( I_{A1} = 0 \). This property is due to the fact that Encoder 1 is a systematic encoder. Even if Decoder 1 is not able to extract any information out of the observations of the parity bits (which is the case for zero a-priori information), the information about the information bits is at least as good as the observation of the systematic bits from the channel outputs.
As opposed to that, the EXIT function $T_2$ of Decoder 2 starts in the origin; i.e., for zero a-priori information, $I_{A2} = 0$, Decoder 2 produces zero extrinsic information, $I_{E2} = 0$. This is a typical property of non-systematic recursive encoders, like Encoder 1 here. If both encoders were non-systematic, both EXIT functions would start in the origin\(^8\), and the iterative decoding process could not start. In the present example, Decoder 1 is fully systematic, and the observations of these systematic bits provide a seed for the iterative decoding process.

For SNR smaller than $E_s/N_0 = -4.75$ dB, the EXIT functions of the two decoders intersect. Correspondingly, the decoding trajectory will get stuck and error-free decoding is impossible. On the other hand, for SNR larger than $E_s/N_0 = -4.75$ dB, there is an open tunnel between the two EXIT functions. Therefore the decoding trajectory will proceed to the upper right corner and error-free decoding is possible if the number of iterations is sufficiently large. The critical SNR value of $E_s/N_0 = -4.75$ dB is referred to as the decoding threshold.

From the EXIT chart, we expect the bit error rate (BER) to be relatively high for SNR below the decoding threshold and very low for SNR above the decoding threshold. In fact for the asymptotic case, where the code length goes to infinity, there is a sharp transition of the BER from 1/2 to 0 at the decoding threshold. For finite code lengths, this threshold is less pronounced but still provides a good guideline for the code design [5].

The concepts introduced are now applied for code design. Assume that we have a communication channel with a single parameter (like the BI-AWGNC with parameter SNR or the BEC with parameter erasure probability) and a set of component codes for the PC code. Then two typical problems for code design are:

**Code design problem 1:** Given a fixed communication channel, find the component codes such that the rate of the PC code is maximised, subject to the constraint that the iterative decoder of the PC code converges.

**Code design problem 2:** Given a fixed code rate for the PC code, find the component codes such that the iterative decoder of the PC code converges for the worst possible communication channel.

Problem 1 is addressed in Section 4 for LDPC codes. Here we focus on Problem 2.

We consider the same rate $1/3$ turbo code as in the previous section, but now with a set of options for the component codes. The generators for the two recursive convolutional encoders are listed in Table 2 (taken from [5]), and they are assumed to have the same memory length (generalisation is possible).

The corresponding EXIT chart is depicted in Fig. 9 for an SNR of $E_s/N_0 = -4.75$ dB. We see that the EXIT functions look very different for different memory lengths. For larger memory lengths, the EXIT functions typically open up towards higher a-priori information

\(^8\)As mentioned in Footnote 2, some authors prefer a “symmetric” representation of the turbo code. This leads then to symmetric EXIT charts, where both EXIT functions start in the origin. Due to the different interpretation, the iterative decoder does still work in this case.
Encoder 1: $g_1$  Encoder 2: $g_2$  Memory length: $m$

\begin{tabular}{|c|c|c|}
\hline
(1, $5/7$) & (5/7) & 2 \\
(1, 17/15) & (17/15) & 3 \\
(1, 35/23) & (35/23) & 4 \\
\hline
\end{tabular}

Table 2: Generators for the component codes of the PC code (in octal notation).

(upper right part of the EXIT chart), as can be observed in the figure; furthermore their decoding threshold is typically at higher SNR. However, one need to be careful as low memory does not necessarily mean low decoding threshold, and careful analysis is required.

![EXIT chart for PC codes](image)

Figure 9: Turbo code (PC code) EXIT functions for component encoders with different memory lengths $m$ (see Table 2) at $E_s/N_0 = -4.75$ dB.

It can be shown that the area between the two EXIT functions corresponds to the difference between the rate of the overall code and the capacity of the communication channel, referred to as the rate loss. Therefore it is desirable to have a good fit of the two EXIT functions and thus a small rate loss. This area property is explained for LDPC codes in Section 4. For details about the area property for PC codes, we refer the reader to [10, 20].

Applying the two criteria (i) open tunnel and (ii) small area between the two EXIT functions, the EXIT chart allows to pick the best component codes from a set of options. This way the best PC code out of a set of candidates can easily be identified, by only investigating the component codes without actually running the iterative decoder for the overall code.
3 Serially concatenated codes

This section addresses the analysis and design of serially concatenated codes. Similarly to the previous section, we review briefly the encoder and decoder structures, and then look into the EXIT analysis, which is similar to that for parallel concatenated codes.

3.1 Encoder and decoder

Consider a serial concatenated (SC) code. The SC encoder is depicted in Fig. 10, and it is defined by two component encoders and an interleaver \( \pi \). The input to Encoder 1 (also called the outer encoder) is the original information word \( u_1 = u \), of length \( K_1 = K \), and the output is the codeword \( x_1 \) of length \( N_1 \). The input to Encoder 2 (also called the inner encoder) is the interleaved codeword, \( u_2 = \mathrm{perm}_\pi x_1 \), of length \( K_2 = N_1 \), and the output is the codeword \( x_2 \) of length \( N_2 \); this is also the SC codeword \( x \). Thus the SC encoder maps the information word \( u \) of length \( K \) to the codeword \( x = x_2 \) of length \( N = N_2 \).

An example for a SC code is a serial Turbo code of rate 1/4, where Encoder 1 is a convolutional encoder of rate 1/2 (not necessarily recursive or systematic), and Encoder 2 is a recursive systematic\(^9\) convolutional encoder of rate 1/2.

The codeword \( x \) is transmitted over the communication channel, yielding the observation vector \( y \). As this corresponds to the noisy observation of only \( x_2 \), we write also \( y_2 = y \). The observation \( y_2 \) is fed to Decoder 2.

The iterative decoder for the SC code is shown in Fig. 11. As for the PC codes, we assume that both component decoders are APP decoders producing extrinsic probabilities\(^10\). While the component decoders of the PC code exchange probabilities of information bits,

\(^9\)Recursive is required for good distance properties and low decoding thresholds, and at least partially systematic \([29]\) or equivalently code doping \([30,31]\) is required to start the iterative decoding process.

\(^{10}\)See also Footnote 4.
the component decoders of the SC code exchange probabilities of the code bits produced by Encoder 1.

Decoder 2 computes the vector of extrinsic values $e_2$ based on the observation $y_2$ and the a-priori values $a_2$ coming from Decoder 1. At iteration $l$, $l = 1, 2, \ldots, L$, we have

$$e^{(l)}_{2k} := P(U_{2k} = 0 | y_2, a^{(l-1)}_{2 \setminus k}),$$

$k = 1, 2, \ldots, K_2$ (remember that $K_2 = N_1$). As in the first iteration, no a-priori values are available, we define $a^{(1)}_2 = [1/2 \ldots 1/2]$. De-interleaving $e^{(l)}_2$ gives the a-priori values

$$a^{(l)}_1 = \text{perm}_{\pi^{-1}} e^{(l)}_2$$

for Decoder 1. Then Decoder 1 computes the vector of extrinsic values $e_1$ based on the a-priori values $a_1$ coming from Decoder 2. At iteration $l$, we have

$$e^{(l)}_{1n} := P(X_{1n} = 0 | a^{(l)}_1),$$

$n = 1, 2, \ldots, N_1$. Interleaving $e^{(l)}_1$ gives the a-priori values

$$a^{(l+1)}_2 = \text{perm}_{\pi} e^{(l)}_1$$

for Decoder 2. After $L$ iterations (or after a certain stopping criterion is fulfilled), Decoder 1 computes the estimate $\hat{u}$ of the information word:

$$\hat{u}_k = \begin{cases} 0 & \text{if } P(U_k = 0 | a^{(L)}_1) \geq 1/2, \\ 1 & \text{otherwise}, \end{cases}$$

$k = 1, 2, \ldots, K$. As for PC code, all computations are typically performed on the code trellis by a forward-backward algorithm [26–28].

### 3.2 EXIT analysis

We investigate now the information transfer between the two component decoders. The a-priori and extrinsic information for Decoder 1 are given by

$$I_{A1} := \frac{1}{N_1} \sum_{n=1}^{N_1} I(X_{1n}; A_{1n}) \quad I_{E1} := \frac{1}{N_1} \sum_{n=1}^{N_1} I(X_{1n}; E_{1n});$$

and the ones for Decoder 2 are given by

$$I_{A2} := \frac{1}{K_2} \sum_{k=1}^{K_2} I(U_{2k}; A_{2k}) \quad I_{E2} := \frac{1}{K_2} \sum_{k=1}^{K_2} I(U_{2k}; E_{2k}).$$

As before, we have $I_{E1} = I_{A2}$ and $I_{A1} = I_{E2}$.

To determine the EXIT functions for Decoder 1 and Decoder 2, we use the two decoding models depicted in Fig. 12 and Fig. 13. While the decoding model for Decoder 2

© Ingmar Land, October 2, 2013 16
Decoder 1 receives no observations from the communication channel, and therefore the communication channel is not used in the decoding model; the dashed box is still included (without connections to encoder or decoder) to show the general structure of the decoding model. Decoder 1 only obtains a-priori values $a_1$ for its codeword $x_1$. For a given model of the a-priori channel, the EXIT function of Decoder 1 is defined as

$$T_1 : [0, 1] \rightarrow [0, 1]$$
$$I_{A1} \mapsto I_{E1} = T_1(I_{A1}).$$  \hfill (27)

Note that this EXIT function is independent of the communication channel.

Decoder 2 receives observations $y_2$ of the codeword $x_2$ from the communication channel and a-priori values $a_2$ for the information word $u_2$. (This is the same decoding model as the one for the component decoders of a PC code.) For a given model of the communication channel and a given model of the a-priori channel, the EXIT function of Decoder 2 is defined as

$$T_2 : [0, 1] \rightarrow [0, 1]$$
$$I_{A2} \mapsto I_{E2} = T_2(I_{A2}).$$  \hfill (28)

Note that this EXIT function depends on the communication channel, as opposed to the EXIT function for Decoder 1.
Figure 14: EXIT functions for a SC code, including the decoding trajectory.

Typical EXIT functions for the component decoders of an SC code, including the decoding trajectory, are depicted in Fig. 14. The EXIT function $T_1$ of Decoder 1 starts at the origin: Decoder 1 does not obtain observations from the communication channel, and accordingly for zero a-priori information at the input, it produces also zero extrinsic information at the output. As opposed to that, the EXIT function $T_2$ of Decoder 2 does not start at the origin: Decoder 2 obtains channel observations, and thus even for zero a-priori information, it produces positive (non-zero) extrinsic information. Without this property\(^{11}\) of the EXIT function of Decoder 2, the iterative process would not get started.

Successful iterative decoding requires three properties of the two EXIT functions:

1. At least one of the two EXIT functions must have non-zero extrinsic information for zero a-priori information. Otherwise, the iterative decoding process can not start.

2. There must be an open tunnel between the two EXIT functions. Otherwise, the iterative process gets stuck before error-free decoding.

3. The number of iterations must be large enough\(^{12}\) such that the decoding trajectory can reach the upper right corner\(^{13}\) ($I_{E1} = 1$ and $I_{E2} = 1$). Otherwise the final decisions are not error-free.

\(^{11}\)For this property of the EXIT function $T_2$, Encoder 2 needs to be systematic or at least partially systematic, i.e. the codeword $x_2$ needs to include at least some bits of the information word $u_2$ [29, 31].

\(^{12}\)Asymptotically, the required number of iterations may go to infinity.

\(^{13}\)In fact, reaching the upper border ($I_{E1} = 1$) or the right border ($I_{E2} = 1$) is sufficient.
The EXIT functions in Fig. 14 fulfil all these properties.

In the following section we show by an example how to design an SC code based on EXIT charts.

### 3.3 Code analysis and design

As an example of a family of SC codes, consider the following: assume for Encoder 1 a convolutional encoder (not necessarily recursive) of rate 1/2 defined by the generator $g_1(D)$ and for Encoder 2 a recursive systematic convolutional encoder of rate 1/2 defined by the generator $g_2(D)$, where

$$g_1(D) = \left(1, \frac{f_1(D)}{r_1(D)}\right) \quad g_2(D) = \left(1, \frac{f_2(D)}{r_2(D)}\right).$$

As before for the PC code, the polynomials are specified in the common octal notation (see Table 3). The interleaver is chosen randomly. Assume that the communication channel is a BI-AWGNC with SNR $E_s/N_0$, and that the a-priori channels are modelled by a BI-AWGNC.

Encoder 1: $g_1$

<table>
<thead>
<tr>
<th>Memory length: $m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5, 7)</td>
</tr>
<tr>
<td>(15, 17)</td>
</tr>
<tr>
<td>(23, 35)</td>
</tr>
</tbody>
</table>

Table 3: Generators for Encoder 1 of the SC code (in octal notation).

There are various problems that can be defined for the design of an SC code. As an example, we consider the following:

**Code design problem 3:** Given Encoder 2, find Encoder 1 such that the iterative decoder of the SC code converges at the lowest possible SNR of the communication channel.

The generator for Encoder 1 may be any of the one given Table 3 (optimum free distance codes, [32]); and we fix the generator of Encoder 2 to $g_2 = (1, 5/7)$.

The EXIT chart is depicted in Fig. 15. Consider first Encoder 1 (also referred to as the outer code). Larger memory lengths lead usually to a steeper slope of the EXIT function for medium a-priori information. This makes it harder to match them to the EXIT function of Encoder 2. Therefore for a low decoding threshold (and thus error-free decoding at small SNR), an Encoder 1 with low memory length is preferable. Note, however, that higher memory lengths lead to lower error floors for finite-length codes. This trade-off needs to be taken into account when designing codes for a practical system [29, 33].

Consider now Encoder 2 (also referred to as the inner code), who’s EXIT function depends on the SNR of the communication channel. As expected, the EXIT functions...
intersect for low SNR and there is an open tunnel for higher SNR. The exact value of the decoding threshold depends on the chosen generator for Encoder 1. From the EXIT chart we see that Encoder 1 with memory length two has the lowest decoding threshold for our example, which is at about $E_s/N_0 \approx -5.9$ dB. For further optimisation, we may also vary the generators of Encoder 2.

![EXIT chart for a SC code](image)

Figure 15: EXIT chart for a SC code. Encoder 1 (outer code, in colour) is taken from Table 3. Encoder 2 (inner code, in black) is the memory-2 encoder from Table 2. The SNR values $E_s/N_0$ of the communication channel are indicated in steps of 0.5 dB.

Serially concatenated codes, with encoder and decoder as shown in Fig. 10 and Fig. 11 may serve as a template for other systems with receivers that operate in an iterative manner. A few important examples are discussed in Section 5.

4 LDPC codes

For PC codes and SC codes, the structure of the encoders is very similar to the structure of the decoders. Correspondingly, the decoding model for the EXIT analysis of the component decoders is very intuitive. This is different for LDPC codes: the encoding algorithm and the iterative decoding algorithm are not much related. Therefore the decoding models for the EXIT analysis of LDPC codes are directly derived from the decoding algorithms. This is done in the following section.

Analysis and design are first treated for the case where the communication channel is a BEC. In this case exact analytical expressions of the EXIT functions are available,
which allows to show how to design codes without any effects from approximations. This design methodology is then adapted to the BI-AWGNC, for which efficient analytical approximations for the EXIT functions are available. We assume that the reader is familiar with regular and irregular LDPC codes, efficient encoding, and iterative decoding [2, 20, 23, 34–38].

4.1 Decoder and decoding models

Assume an irregular LDPC code of rate $R$ with degree polynomials $\lambda(z)$ and $\rho(z)$ for the variable nodes and the check nodes (from edge perspective), respectively,

$$\lambda(z) = \sum_i \lambda_i z^{i-1}, \quad \rho(z) = \sum_i \rho_i z^{i-1},$$

where $\lambda_i$ denotes the relative number of edges connected to variable nodes of degree $i$, and $\rho_i$ denotes the relative number of edges connected to check nodes of degree $i$. Denote further $\bar{d}_v$ the average variable-node degree and $\bar{d}_c$ the average check-node degree. The rate $R$ of the code, the design rate $R_d$, and the degree coefficients are related by

$$R \geq R_d = 1 - \frac{\sum_i \rho_i / i}{\sum_i \lambda_i / i} = 1 - \frac{\bar{d}_v}{\bar{d}_c}, \quad (29)$$

see e.g. [20]. In practice, the actual rate $R$ and the design rate $R_d$ usually differ only slightly, and we may ignore this subtlety in the following.

Variable-node decoder

Consider the decoder for a variable node of degree $d_v$ with associated bit $x$, as depicted in Fig. 16. The decoder receives the observation $y$ from the communication channel and the a-priori values $a_{v,j}$, $j = 1, 2, \ldots, d_v$, and it computes the extrinsic probabilities

$$e_{v,j} = P(X = 0 | y, a_{v,\backslash j}), \quad (30)$$

![Figure 16: Decoder for variable-node of degree $d_v$.](image)
The a-priori values may be modelled as observations of repetitions of \( x \) via the a-priori channel. Correspondingly, the decoding model for the variable-node decoder includes a repetition code of length \( d_v \), as depicted in Fig. 17. For convenience define the codeword produced by this repetition encoder as \( x_v = [x_{v1} \ldots x_{vd_v}] \); note that \( x = x_{v1} = \cdots = x_{vd_v} \). Define the a-priori information and the extrinsic information as

\[
I_{Av} := \frac{1}{d_v} \sum_{j=1}^{d_v} I(X_{vj}; A_{vj}) \\
I_{Ev} := \frac{1}{d_v} \sum_{j=1}^{d_v} I(X_{vj}; E_{vj}).
\]

(31)

The EXIT function for the variable-node decoder is then defined as

\[
T_{v,d_v} : [0,1] \rightarrow [0,1] \\
I_{Av} \mapsto I_{Ev} = T_{v,d_v}(I_{Av}).
\]

(32)

Note that this EXIT function depends on the communication channel (as the decoding model includes the communication channel) and on the variable node degree.

![Diagram of Decoding Model](image)

Figure 17: Decoding model for variable-node decoder.

The overall EXIT function of the variable-node decoder is obtained by averaging with respect to the variable-node degrees:

\[
T_v(I_{Av}) = \sum_\lambda \lambda I_{Av} = T_{v,d_v}(I_{Av}).
\]

(33)

Note that in every iteration step all variable node decoders obtain the same a-priori information, and thus see the same a-priori channel.

For the case where the a-priori channel is a BEC, the EXIT functions have a special area property. Denote \( I_{ch} \) the symmetric capacity\(^{14} \) of the communication channel, and denote the area under the variable-node EXIT function by

\[
\mathcal{A}_v = \int_0^{I_{ch}} T_v(I_{Av}) dI_{Av}.
\]

Then we have

\[
\mathcal{A}_v = 1 - \frac{1 - I_{ch}}{d_v},
\]

(34)

where \( \bar{d}_v \) is the average variable-node degree, as defined before. This equation relates the area under the EXIT function to the average degree \( [10] \).

\(^{14}\)The symmetric capacity of a channel is the mutual information between channel input and channel output for a uniform input distribution.
Consider the decoder for a check node of degree $d_c$ with associated bits $x_{cj}$, $j = 1, 2, \ldots, d_c$, as depicted in Fig. 18. The decoder does not receive any observations from the communication channel, but it obtains one a-priori value $a_{cj}$ for every bit $x_{cj}$, $j = 1, 2, \ldots, d_c$; and it computes the extrinsic probabilities
\[
e_{cj} = P(X_{cj} = 0 | a_{cj}),
\]
$j = 1, 2, \ldots, d_c$. The bits $x_{cj}$ fulfill the parity check equation $x_{c1} \oplus x_{c2} \oplus \cdots \oplus x_{c,d_c} = 0$, and thus form the codeword $x_c = [x_{c1} \ldots x_{c,d_c}]$ of a single-parity-check (SPC) code. Correspondingly, the decoding model for the check-node decoder includes a SPC code of length $d_c$, as depicted in Fig. 19. With the a-priori information and the extrinsic information
\[
I_{Ac} := \frac{1}{d_c} \sum_{j=1}^{d_c} I(X_{cj}; A_{cj}), \quad I_{Ec} := \frac{1}{d_c} \sum_{j=1}^{d_c} I(X_{cj}; E_{cj}),
\]
respectively, the EXIT function for the check-node decoder is defined as
\[
T_{c,d_c} : [0, 1] \rightarrow [0, 1], \quad I_{Ac} \mapsto I_{Ec} = T_{c,d_c}(I_{Ac}).
\]
This EXIT function depends on the check-node degree, but not on the communication channel.

The overall EXIT function of the check-node decoder is obtained by averaging with respect to the check-node degrees:
\[
T_c(I_{Ac}) = \sum_i \rho_i T_{c,i}(I_{Ac}).
\]
Similarly to the VN decoders, in every iteration step all check node decoders obtain the same a-priori information, and thus see the same a-priori channel.
Figure 19: Decoding model for check-node decoder.

When the a-priori channel is a BEC, the EXIT functions again fulfill an *area property*. Denote the area under the check-node EXIT function by

\[ A_c = \int_0^1 T_c(I_{Ac}) \, dI_{Ac}. \]

Then we have

\[ A_c = \frac{1}{\bar{d}_c}, \quad (39) \]

where \( \bar{d}_c \) is the average check-node degree, as defined above. This equation again relates the area under the EXIT function to the average degree [10].

Discussion of decoding models

The decoding models used for the PC codes, the SC codes\(^{15}\) and the LDPC codes are quite different. Depending on the decoding situation, the bits transmitted over the communication channel and the a-priori channel may be code bits or information bits; moreover, bits may be transmitted over the communication channel or not. For other code structures, the decoding may be extended to the general case where two different encoders are used for the communication channel and the a-priori channel. In fact, the decoding models used in this chapter are special cases of this general decoding model [10].

Discussion of the area properties

The two area properties (34) and (39) have important implications on the code design. Successful decoding requires that the variable-node EXIT function lies above the check-node EXIT functions (see e.g. Fig. 20 or Fig. 21). A necessary condition for that is that the area above the variable-node EXIT function, \( 1 - A_v \), is smaller than the area under the check-node EXIT function\(^{16}\), \( A_c \). We thus thus require

\[ 1 - A_v < A_c. \quad (40) \]

\(^{15}\)Only the decoding model for Encoder 1 (outer encoder) of the SC code is different from the decoding models of the component decoders of the PC code.

\(^{16}\)Remember that the axes are swapped for the check-node EXIT function.
This condition has two important implications.

First we substitute (34) and (39) in this inequality, use (29), and obtain

\[ R_d = 1 - \frac{\bar{d}_e}{d_c} < I_{ch}. \] (41)

Thus, the rate can not exceed the channel capacity if the two EXIT functions do not intersect. This agrees with the channel coding theorem [24], as expected. Furthermore, equality of code rate and capacity requires equality in (40).

Second, assume that \( 1 - A_v = \gamma A_c \) for some \( \gamma \in [0, 1) \). Note that the closer \( \gamma \) is to 1, the closer the rate is to the channel capacity. Using, (29), (34) and (39), we can relate the rate \( R_d \), the capacity \( I_{ch} \) of the communication channel and the value \( \gamma \) as

\[ R_d = \frac{I_{ch} - (1 - \gamma)}{\gamma} < I_{ch}. \] (42)

As can be seen from this inequality, the larger the gap between the two EXIT functions (i.e., the smaller \( \gamma \)), the more the rate \( R_d \) is bounded away from the capacity \( I_{ch} \). Therefore, a code rate close to capacity requires that the gap between the two EXIT functions vanishes. In other words: the two EXIT functions have to be matched to maximise the code rate. The code design problem thus reduces to a curve fitting problem.

### 4.2 Analysis and design for the BEC

When the communication channel is a BEC, the a-priori channel may be modelled as a BEC without any approximations, and then the EXIT analysis provides exact results. Even more, the EXIT functions for the variable-node decoder and for the check-node decoder can be expressed in closed form. This section deals with these relationships and shows how to formulate the code design as a particularly simple optimisation problem, namely a linear program, which can easily be solved.

The BEC has the input alphabet \( \{0, 1\} \), the output alphabet \( \{0, 1, \Delta\} \). The output value is the erasure symbol \( \Delta \) with probability \( \delta \), and it is equal to the input value with probability \( 1 - \delta \). Furthermore, for uniform input, the mutual information between input \( X \) and output \( Y \) is

\[ I(X; Y) = 1 - \delta, \] (43)

which is also the capacity of the BEC.

**EXIT functions**

Consider first the decoding model for a variable-node decoder with degree \( d_v \), as depicted in Fig. 17. Denote \( \delta_{ch} \) the erasure probability of the communication channel, \( \delta_{Av} \) the erasure probability of the a-priori channel, and \( \delta_{Ev} \) the erasure probability for the extrinsic values. The extrinsic message \( e_{vj} \) at the variable-node decoder output is an erasure if all incoming messages \( y \) and \( a_{vk}, k \neq j \), are erasures. Thus we have

\[ \delta_{Ev} = \delta_{ch} \cdot \delta_{Av}^{d_v - 1}. \]
Using the relationship between mutual information and erasure probability from (43), we obtain the EXIT function of the variable-node decoder for a node of degree $d_v$ as

$$I_{Ev,d_v} = T_{v,d_v}(I_{Av})$$

$$= 1 - (1 - I_{ch})(1 - I_{Av})^{d_v-1}.$$  

Using (33), the overall EXIT function for the variable-node decoders is then given by

$$I_{Ev} = T_v(I_{Av}) = \sum_i \lambda_i \left[ 1 - (1 - I_{ch})(1 - I_{Av})^{d_v-1} \right]$$

$$= 1 - (1 - I_{ch}) \cdot \sum_i \lambda_i (1 - I_{Av})^{d_v-1}. \quad (44)$$

The individual EXIT functions as well as the average EXIT function start at $I_{Ev} = I_{ch}$ for $I_{Av} = 0$ and end at $I_{Ev} = 1$ for $I_{Av} = 1$, where $I_{ch} = 1 - \delta_{ch}$ is the mutual information of the communication channel (channel capacity). The variable-node EXIT functions for some variable-node degrees are depicted in Fig. 20.

Figure 20: EXIT functions of variable-node decoders for degrees $d_v \in \{3, 6, 12\}$, and EXIT functions of check-node decoders for degrees $d_c \in \{2, 4, 6\}$. The communication channel has the erasure probability $\delta_{ch} = 1/4$, and thus capacity $I_{ch} = 3/4$.

Consider now the decoding model for a check-node decoder with degree $d_c$, as depicted in Fig. 19. Denote $\delta_{Ac}$ the erasure probability of the a-priori channel, and $\delta_{Ec}$ the erasure probability for the extrinsic values. The extrinsic message $e_{cj}$ at the check-node decoder

© Ingmar Land, October 2, 2013 26
output is not an erasure if all incoming messages $a_{ck}$, $k \neq j$, are not erasures. Thus we have

$$1 - \delta_{Ec} = (1 - \delta_{Ac})^{d_c-1}.$$  

Using the relationship between mutual information and erasure probability from (43), we obtain the EXIT function of the check-node decoder for a node of degree $d_c$ as

$$I_{Ec,d_c} = T_{c,d_c}(I_{Ac}) = I_{Ac}^{-1}.$$  

Using (38), the overall EXIT function for the check-node decoders is then given by

$$I_{Ec} = T_{c}(I_{Ac}) = \sum_i \rho_i I_{Ac}^{i-1}. \tag{45}$$  

The individual EXIT functions as well as the average EXIT function start at $I_{Ec} = 0$ for $I_{Ac} = 0$ and end at $I_{Ec} = 1$ for $I_{Ac} = 1$. The check-node EXIT functions for some check-node degrees are depicted in Fig. 20.

Note the axis labels in Fig. 20: for the variable-node EXIT functions, the a-priori information is on the x-axis and the extrinsic information on the y-axis; for the check-node EXIT functions, the a-priori information is on the y-axis and the extrinsic information on the x-axis. This is useful, as the extrinsic information of the variable-node decoders becomes the a-priori information for the check-node decoders, and vice versa.

**Code design**

The EXIT functions derived above are now applied to design LDPC codes. A typical question to be asked is as follows:

**Code design problem 4:** Given a family of LDPC codes and a fixed communication channel, find the degree distributions that maximise the code rate.

We assume that the check-node distribution is fixed and that the variable-node degrees are upper-bounded by $d_{v,max}$ to limit decoding complexity.

For a fixed check-node distribution, the design rate is maximal if $\sum_i \lambda_i / i$ is maximal, by (29). This is the objective function of our optimisation problem. The constraints are as follows:

1. We want that the iterative decoder converges. Therefore there needs to be an open tunnel between the variable-node EXIT function and the check-node EXIT function.

2. The coefficients $\lambda_i$ have to be chosen such that they represent a valid degree distribution.

3. The stability condition for the BEC,

$$\delta_{ch} \cdot \chi'(0) \cdot \rho'(1) < 1, \tag{46}$$

has to be fulfilled to ensure convergence in the upper right corner of the EXIT chart and to thus avoid error floors [20].
Given the EXIT function of the check-node decoder (which is fixed for this problem) as
\[ T_c(I) = \sum_i \rho_i I^{i-1}, \]
define its inverse as
\[ g(I) = T_c^{-1}(I). \] (47)

Then above problem can be formulated as the following optimisation problem:

\[
\begin{align*}
\text{maximise} & \quad \sum_{i=2}^{d_v,\text{max}} \frac{\lambda_i}{i} \\
\text{subject to} & \quad 1 - (1 - I_{ch}) \cdot \sum_i \lambda_i (1 - I)^{d_v - 1} \geq g(I) \quad \text{for } I \in [0, 1] \\
& \quad \sum_{i=2}^{d_v,\text{max}} \lambda_i = 1 \\
& \quad 0 \leq \lambda_i \leq 1 \quad \text{for } i = 2, 3, \ldots, d_v,\text{max} \\
& \quad \lambda_2 < 1/((\delta_{ch} \cdot \sum_i (i - 1)\rho_i)).
\end{align*}
\] (48) (49) (50) (51) (52)

The objective function and the constraint functions are all linear in the coefficients \( \lambda_i \), and therefore this optimisation problem is a linear program, for which efficient solvers exist [39].

As an example we consider LDPC codes with variable-node degrees between 2 and 10 and a fixed check-node degree of 6. We further assume that the communication channel has an erasure probability of \( \delta_{ch} = 0.4 \) and thus a capacity of \( C = 0.6 \) (which is the maximal rate we can achieve). Solving the problem above for these parameters yields the variable-node degree coefficients as given in Table 4 and the design rate \( R_d = 0.56 \). The corresponding EXIT functions are depicted in Fig. 21.

For the given check node degree of \( d_c = 6 \), variable-node degrees higher than 5 are not required to match the two EXIT functions. From Fig. 21 there is a gap only at the beginning of the iterations. To further close this gap and thus increase the code rate, a higher check-node degree is required. In this case, the optimisation would also yield higher variable-node degrees.

<table>
<thead>
<tr>
<th>( \lambda_2 )</th>
<th>( \lambda_3 )</th>
<th>( \lambda_4 )</th>
<th>( \lambda_5 )</th>
<th>( \lambda_6 )</th>
<th>( \lambda_7 )</th>
<th>( \lambda_8 )</th>
<th>( \lambda_9 )</th>
<th>( \lambda_{10} )</th>
<th>( \rho_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.500</td>
<td>0.117</td>
<td>0.310</td>
<td>0.072</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 4: Degree coefficients of optimised LDPC code.

### 4.3 Analysis and design for the AWGN channel

If the communication channel is not a BEC, the a-priori channel is usually modelled by a BI-AWGNC. In this case, the EXIT analysis does not provide exact results but still very
good approximations. In this section, the approach is explained for the case where the communication channel is also a BI-AWGNC\textsuperscript{17}.

For the BEC, there is a simple relationship between the mutual information of the channel and the parameter of the channel (the erasure probability $\delta_{ch}$). For the BI-AWGNC there is a similar relationship between the mutual information between channel input and channel output and the standard deviation of the L-values at the channel outputs.

Assume a BI-AWGNC: the code bits $X \in \{0,1\}$ are mapped to $\{+1,-1\}$, and then white Gaussian noise with variance $\sigma_w^2 = N_0/2E_s$ is added, yielding the output $Y \in \mathbb{R}$; $E_s/N_0$ denotes the SNR. The L-value of the channel output is
\[ l = \log \frac{P(y|X=0)}{P(y|X=1)} = \frac{4E_s}{N_0} \cdot y. \]

For a given $X$, $Y$ is Gaussian and so is $L$. Assume now that $X = 0$ is given ($X = 1$ works equivalently). A Gaussian L-value has the property that its conditional mean value $\mu_L = 4E_s/N_0$ (given $X = 0$) and its conditional variance $\sigma_L^2 = 8E_s/N_0$ (given $X = 0$) are related as $\sigma_L^2 = 2\mu_L$. Thus the statistical properties of a Gaussian L-value are completely characterised by the conditional standard deviation $\sigma_L$ (given $X = 0$). We now define the

\textsuperscript{17}For communication channels that are not BI-AWGNC, the EXIT functions for variable-node decoders may be determined by simulation of the decoding model, such that they are numerically available for the code optimisation.
J-function as

\[ I(X; L) = J(\sigma_L), \]  

(53)

and we denote its inverse by \( J^{-1}(I) \). This function may be numerically evaluated and stored in a look-up table; alternatively, it can be numerically approximated, e.g. [8, 40]. The J-function is used to describe the EXIT-functions of the variable-node decoder and the check-node decoder.

Consider a variable-node of degree \( d_v \). Assume that all messages are L-values. Then the variable-node decoder simply adds up the corresponding messages to obtain the extrinsic message. Adding Gaussian random variables gives again a Gaussian random variable, and thus the extrinsic L-value is Gaussian (all conditioned on \( X = 0 \)), where the variance is the sum of the variances of the incoming messages. Using the J-function from (53) for conversion between mutual information and standard deviation, we obtain the EXIT function for a variable-node decoder of degree \( d_v \) as [8]

\[ I_{Ev,d_v} = T_{v,d_v}(I_{Av}) = J\left(\sqrt{\sigma_{ch}^2 + (d_v - 1)(J^{-1}(I_{Av}))^2}\right), \]  

(54)

where \( \sigma_{ch} = 8E_s/N_0 \) denotes the standard deviation of the L-values of the communication channel. Note that this expression is exact. The average EXIT function of the variable-node decoders is obtained by substituting this expression in (33).

Consider now a check-node decoder of degree \( d_c \). When there is no communication channel and the a-priori channel is a BEC, the EXIT function \( T_C(.) \) of a code and the EXIT function \( T_{C\perp}(.) \) of its dual code are related by the duality theorem [10]:

\[ T_{C\perp}(I_A) = 1 - T_C(1 - I_A). \]

This relationship holds approximately also for the BI-AWGNC. Noting that the repetition code and the single-parity-check code are dual codes, the EXIT function for a check-node decoder of degree \( d_c \) is with good approximation given by [8]

\[ I_{Ec,d_c} = T_{c,d_c}(I_{Ac}) = 1 - J\left(\sqrt{(d_c - 1)(J^{-1}(1 - I_{Ac}))^2}\right). \]  

(55)

The average EXIT function of the check-node decoders is obtained by substituting this expression in (38).

Given (54) and (55), the approach for code design is identically to the one for the BEC, as given in the previous section; just the EXIT functions need to be replaced. This approach has been followed in [8, 9] for the design of coded modulation with LDPC codes and IRA codes.

5 Comments and generalisations

This chapter has given an introduction to the EXIT chart method for analysis and design of parallel concatenated codes, serially concatenated codes, and for LDPC codes. A few comments on this method, generalisations and further applications are discussed in the following.
5.1 Estimation of mutual information

The EXIT analysis is valid only if the component decoders perform true APP decoding. To show that, assume that a decoder computes the correct probability, but this probability is then distorted by a bijective function\(^{18}\). As the function is bijective, the mutual information between the bit and the wrong “probability value” stays the same, and thus this distortion is not reflected in the EXIT chart. The following component decoder, however, which obtains this wrong probability, interprets it as a true probability, and correspondingly the output of this decoder is completely wrong. Thus, the assumption of APP decoders is mandatory, such that the EXIT analysis reflects the behaviour of the actual iterative decoding process. The only exception are bit-flipping algorithms for LDPC codes with binary messages \([41, 42]\). If sub-optimal decoders are applied for the component codes, the extrinsic values may be passed through correction functions to improve the overall performance and to make the EXIT chart method applicable again \([43, 44]\).

EXIT charts depict the mutual information between a uniformly distributed bit \(X\) and the probability \(19\) of this bit, i.e., the a-priori value \(A\) or extrinsic value \(E\). To compute the mutual information between \(X\) and \(E\) (similarly for \(A\)), there are two common approaches.

In the first approach, sometimes referred to as the “histogram method”, the histograms of \(E\) given \(X = 0\) and given \(X = 1\) are determined by simulation. These histograms then serve as estimates of the conditional distributions \(P_{E|X}\). Based on these estimates, the distribution \(P_E\) is computed (which is then also an estimate). Finally the mutual information \(I(X; E)\) is calculated. The advantage of this method is that it works for any decoder, independently of whether it does optimal APP decoding or not.

The second approach exploits the fact that the decoder outputs are APPs, i.e., that \(P(X = 0|E = e) = e\) by definition of the probability \(e\). Using this property, the mutual information can be written in terms of \(e\):

\[
I(X; E) = \int e P_E(e) \, I(X; E = e) \, de \\
= \int e P_E(e) \left[ H(X) - H(X|E = e) \right] \, de \\
= 1 - \int e P_E(e) \, H_2(e) \, de ,
\]

where we use that \(H(X) = 1\) as \(X\) is uniform, and that \(H(X|E = e) = H_2(e)\); \(H_2(\cdot)\) denotes the binary entropy function. Notice that the integral with respect to \(e\) is simply the expectation with respect to \(e\). Given a set of extrinsic values \(E = \{e_1, e_2, e_3, \ldots\}\), this expression can conveniently be estimated by replacing the expectation with simple averaging:

\[
I(X; E) \approx 1 - \frac{1}{|E|} \sum_{e \in E} H_2(e) .
\]

---

\(^{18}\)If the decoder produces an L-value, this L-value may be multiplied by a constant to obtain such a distortion.

\(^{19}\)In this text we have used probabilities. Use of L-values is equivalent, as there is a one-to-one relationship between probabilities and L-values.
This way, no histograms are required and simple averaging is sufficient. This convenient method was introduced for binary codes in [45] and later extended to non-binary codes in [46]. The only disadvantage of the method is that it relies on correct APPs; on the other hand, only then the EXIT chart method is exact, as discussed above.

5.2 Theory of EXIT analysis

EXIT functions were originally introduced to characterise the behaviour of iterative decoders, and to use curve fitting to design codes. Later on, various theories around EXIT functions were developed, some of which some have been used in this chapter.

If the a-priori channel is a BEC, the area theorem relates the area between the two EXIT functions to the gap between the overall code rate and the capacity of the communication channel. For LDPC codes this has been explained in Section 4. For other codes and further details, we refer the reader to [10]. If in addition, also the communication channel is a BEC, a duality applies for the EXIT function of a code and the EXIT function of its dual code [10]. For other channels, this property holds still approximately and may be used to simplify code design, as applied in Section 4.3. EXIT functions may also be used to relate the performance of codes under belief-propagation to the performance under MAP decoding [20, 47].

If the communication channel is not a BEC, the a-priori channel is often modelled as a BI-AWGNC. In this case, the EXIT chart method provides only approximate results and is not exact. For LDPC codes, the method of information combining has been used to compute tight upper and lower bounds of the EXIT functions for the variable-node decoders and the check-node decoders, assuming only that the a-priori channel is from the family of binary-input symmetric memoryless channels. In fact, these EXIT functions are upper-bounded and lower-bounded by the cases where the a-priori channels are BECs or binary symmetric channels (BSCs), respectively. This way, upper and lower bounds on the decoding thresholds can be computed using the EXIT chart approach. Details on information combining and the application to LDPC codes can be found in [11–14, 48]

5.3 EXIT analysis for other codes or coded systems

This chapter focusses on EXIT charts for parallel concatenated codes, serially concatenated codes and LDPC codes. However, the EXIT chart method has also been applied to other codes or coded systems. Some of those are outlined in the following.

A generalisation of parallel and serial concatenation and the corresponding EXIT analysis was proposed in [33]. Parallel concatenated codes with multiple component codes were analysed in [49, 50]. Irregular repeat accumulate (IRA) codes [3, 4] may be seen as special LDPC codes or as well as special serially concatenated codes. Their EXIT analysis is similar to that of serially concatenated codes and has be addressed in [9, 30, 51]. The

---

20For APP decoders, the two methods yield the same result. This fact may be used to verify that the decoder output is indeed an APP.
specific design with complexity constraint, namely a limited number of decoder iterations, has been dealt with in [52, 53].

The EXIT analysis of non-binary codes is a particular challenge. The messages within the decoder are probability distributions of the code symbols. For binary codes, this probability can be represented by a single number, e.g. the probability for the bit being zero or the L-value of the bit. For non-binary codes, a vector of numbers is required. Correspondingly, the model for the a-priori channel may not only have a single but multiple parameters, which makes the analysis very difficult. Approaches to solve this problem were proposed in [46, 54].

The EXIT chart method usually addresses the asymptotic case where the code length goes to infinity. To apply the method to finite length codes, the EXIT function of the component decoder may be replaced by an EXIT band [55]. For BECs, there are analytical results for LDPC codes [56]. The design of serially concatenated systems for finite block lengths has been addressed in [57].

So far we have discussed only pure channel coding problems. However, there are many iterative receiver structures that can also be analysed with the EXIT chart method. Most of them comprise two serially concatenated receiver stages, and correspondingly their EXIT analysis resembles the analysis of SC codes. One very important application is iterative decoding and equalisation, referred to as turbo equalisation, for coded transmission over communication channels with memory [58, 59]. In coded multiuser systems, iterative decoding and interference-cancellation may be used to maximise performance [60–63]. A third important example is the design of coded modulation systems, where the receiver iterates between a soft-demodulator and a channel decoder [64]; efficient design of modulation with LDPC codes and IRA codes has been proposed in [8, 9]. Information transfer analysis has further been applied for space-time coded modulation [65]. EXIT charts allow also to optimise codes for source coding, like in joint source-channel coding, where the receiver iterates between the source decoder and the channel decoder [66, 67].

In the EXIT chart method, the evolution of the mutual information is tracked through the iterations. Mutual information is only one possible parameter of a probability distribution. Similar methods were proposed, where other parameters are considered. Gallager proposed tracking of the probability of error [41]. Variance of the distribution and signal-to-noise ratio have been applied in [68]. Various parameters were compared in [69], and mutual information has been identified as a robust and thus useful measure.

6 Summary

This chapter has given an introduction into the EXIT chart method for parallel concatenated codes, serially concatenated codes and LDPC codes. The intention is to explain the concepts, provide intuition, evoke interest, and to supply enough links such that the reader can start to explore the vast literature about EXIT charts by themselves.

The presentation of the material has focuses on ideas and concepts rather than technical details and mathematical rigour. Furthermore the list of references is not exhaustive.
and rather provides pointers to the different areas and aspects. We would be very happy to receive any feedback and adapt the material or include further material in future revisions.

References


© Ingmar Land, October 2, 2013
[30] S. ten Brink, “Rate one-half code for approaching the Shannon limit by 0.1 db,”


[33] A. Graell i Amat, L. K. Rasmussen, and F. Brännström, “Unifying analysis and
design of rate-compatible concatenated codes,” IEEE Trans. Commun., vol. 59, no. 2,

[34] T. J. Richardson and R. L. Urbanke, “Efficient encoding of low-density parity-check


[38] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-

2004.

[40] E. Sharon, A. Ashikhmin, and S. Litsyn, “EXIT functions for binary input memory-
2006.

1963.


[43] I. Land, “Reliability information in channel decoding – practical aspects and
information theoretical bounds,” Ph.D. dissertation, University of Kiel, Germany,

[44] G. Lechner and J. Sayir, “Improved sum-min decoding for irregular LDPC codes,”


© Ingmar Land, October 2, 2013 37


