Commun. Fac. Sci. Univ. Ank. Series A2-A3
V. 49(1). pp.9-25. (2005)

A REVIEW OF TURBO CODING AND DECODING

MURAT H. SAZLI

Ankara University, Faculty of Engineering, Department of Electronics Engineering

06100 Tandoğan, Ankara, TURKEY

E-mail: sazli@eng.ankara.edu.tr

(Received Dec. 21, 2004; Accepted April 11, 2005)

ABSTRACT

Turbo coding and decoding have opened a new era in the field of channel coding. In this paper, we present a review of turbo coding and decoding with all the derivations in detail. Also elaborated in the paper is the mostly used decoding algorithm, BCJR (Bahl, Cocke, Jelinek and Raviv) algorithm (also known as Maximum A Posteriori algorithm -MAP algorithm).

KEYWORDS: Turbo coding / decoding, channel coding, BCJR algorithm, MAP algorithm.

1. INTRODUCTION

Error control coding, or *channel coding*, is an essential part of a communication system. By adding redundancy to the message to be transmitted in a systematic manner, it is possible to detect and correct errors, which unavoidably occur during the transmission through the communication channel.

Shannon's 1948 paper entitled "A Mathematical Theory of Communication" [3] has been considered as the birth of a new field, *"error control coding"*. In that paper Shannon defined the concept of *"channel capacity"*. He then showed that there exist error control codes that can yield arbitrarily low errors at the receiver output, so long as the transmission rate through the channel is less than the channel capacity. Although he showed the existence of such codes, he did not specify how to construct them. Therefore, that part has been remained unanswered to the researchers that followed in the coming decades.

Even though many efficient coding and decoding schemes have been developed following Shannon, we have not been able to approach to the channel capacity limit within just a few tenths of a dB until the invention of the state-of-the-

art turbo codes in 1993. In other words, turbo codes have closed the significant gap between the coding gains so far achieved using the conventional coding and decoding schemes, and the Shannon's channel capacity limit.

2. TURBO ENCODER

A turbo encoder consists of a parallel concatenation of two (or more) systematic codes. Each of the encoders/decoders is called as "component encoders/decoders". If there are two component encoders in the turbo encoder, it is referred to as a "*two dimensional turbo code*".

Originally, turbo codes were developed with Recursive Systematic Convolutional (RSC) encoders [1], [2]. In this paper, we review turbo codes with convolutional encoders/decoders as component encoders/decoders. Our purpose is to present turbo codes and their innovative decoding scheme along with its decoding algorithm under the light of recent progresses in a unified manner. Please note that we stick with the notation used in the papers of Berrou et al. ([1], [2]) in which the turbo codes were originally introduced.

It is pointed out in [1] that the bit error rate (BER) of classical Non Systematic Convolutional (NSC) code is lower than that of a classical Systematic code with the same code memory at large signal-to-noise ratios (SNR). However, at low SNR's it is in general the other way around. They also introduced in [1] that Recursive Systematic Convolutional (RSC) codes can be better than the best NSC code at any SNR for high code rates. Now, we will briefly decribe RSC.

2.1 Recursive Systematic Convolutional Codes

Before we present the RSC, let us first review classical Non Systematic convolutional encoder. Consider a binary rate R=1/2 convolutional encoder with constraint length K and memory ν=K-1. The input to the encoder at time k is a bit d_k and the corresponding codeword C_k is the binary couple (X_k, Y_k) with

$$X_k = \sum_{i=0}^{K-1} g_{1i} d_{k-i} \quad \text{mod.2} \quad g_{1i} = 0,1 \tag{1.a}$$

$$Y_k = \sum_{i=0}^{K-1} g_{2i} d_{k-i} \quad \text{mod.2} \quad g_{2i} = 0,1 \tag{1.b}$$

here G_1: $\{g_{1i}\}$, G_2: $\{g_{2i}\}$ are the two code generators, generally expressed in octal form.

NSC defined by code generators G_1=37, G_2=21, with memory ν=4 is given in Figure1.

RSC code obtained from the NSC using a feedback loop is given in Figure 2. Notice that one of the two outputs (in this case X_k) is set equal to the information

bit d_k, therefore making the code systematic. Also notice that for an RSC code, the shift register (memory) input is no longer the information bit d_k but is a new variable a_k. This, of course, is due to the feedback in the encoder structure shown in Figure 2. If $X_k = d_k$, then the output Y_k is equal to (2) by substituting a_k for d_k. a_k is recursively calculated as

$$a_k = d_k + \sum_{i=1}^{K-1} \gamma_i a_{k-i} \quad \text{mod.2} \tag{2}$$

where γ_i is respectively equal to g_{1i} if $X_k = d_k$ and to g_{2i} if $Y_k = d_k$.

In [2], authors showed that variable a_k exhibits the same statistical property with d_k. As a result of this, the trellis structure is identical for the RSC code and the NSC code and these two codes have the same free distance d_f. However, as they pointed out in [1], the two ouput sequences $\{X_k\}$ and $\{Y_k\}$ do not correspond to the same input sequence $\{d_k\}$ for RSC and NSC codes. That is the main difference between the two codes.

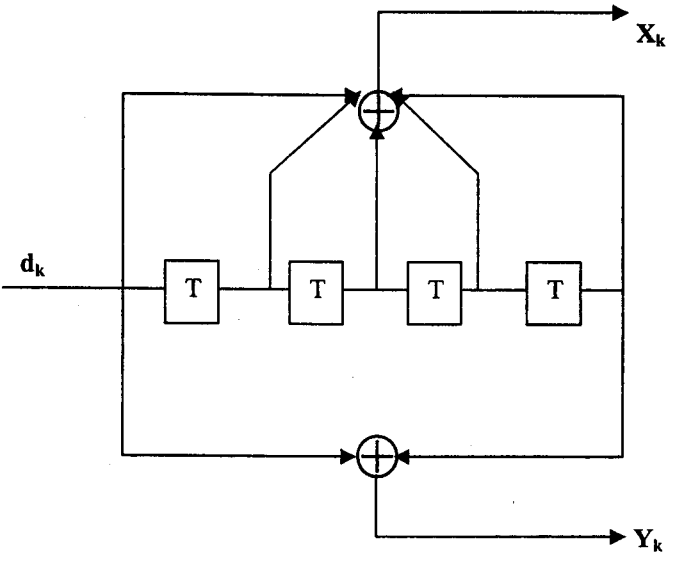

Figure 1. Classical Non Systematic Convolutional Code

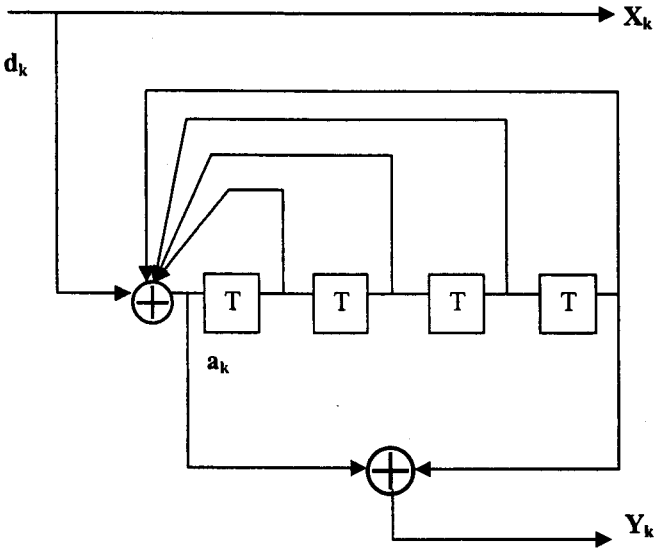

Figure 2. Recursive Systematic Convolutional Code (rate 1/2)

2.2 Parallel Concatenation of RSC Codes

In Figure 3, a rate 1/3 turbo encoder is shown. Data (d_1,d_N) is encoded with blocks. Block size is determined by the size of the interleaver. Original data stream is encoded by the first encoder and produces a parity bit sequence (Y_{11},Y_{1N}). The interleaver shuffles original data stream and then this scrambled version of the data stream is encoded by the second encoder producing the second parity bit sequence (Y_{21},Y_{2N}). Interleaving may be according to a certain rule, or it may be totally random. Several different interleavers have been studied in the literature. According to the results summarized in [4], there is not a significant difference in BER (bit error rate) performance obtained by random interleavers and by the other interleavers. An important consideration, however, is the size of the interleaver. Usually, the bigger is the interleaver size; the better is the performance of turbo decoder. As we will clarify later when we explain the turbo decoder in detail, each

component decoder in the turbo decoder performs a decoding of the same bit d_k, but according to a different sequence due to the interleaving, and thereby providing a relevant piece of information about the accuracy of the decoding made by them to each other in an iterative scheme to improve the decision made by each decoders. Therefore, a bigger block size (or interleaver size) may yield a better separation between the position of a bit in the original data stream and the position of the same bit in the scrambled version, by making the parity bits produced by each encoder "less correlated" when received for decoding. This may be perceived as a "diversity" effect.

Another issue related to turbo encoder is "code puncturing". Parity bits of the both component encoders may be multiplexed with the information bit and transmitted. This yields a rate 1/3 turbo code. Alternatively, parity bits of the component encoders may be multiplexed with the information bit according to a pre-defined pattern. This is called as "code puncturing" and the resultant code is called a "punctured code". For instance, (X_k, Y_{1k}) is transmitted at one signaling interval, and $(X_{k+1}, Y_{2(k+1)})$ is transmitted in the next interval. Therefore, for each information bit, only one parity bit is transmitted, and that results in a rate 1/2 turbo code. Since the parity bit of one component decoder is deleted at each signaling interval, corresponding decoder uses "zero" for that interval during decoding. A detailed discussion on code puncturing may be found in [4].

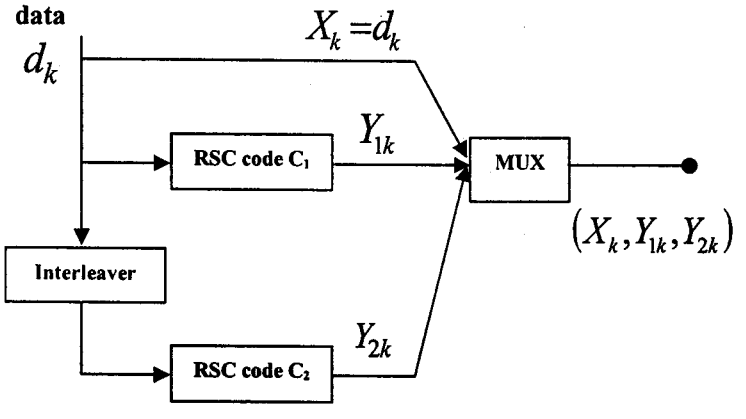

Figure 3. Rate 1/3 turbo encoder

3. TURBO DECODER

To facilitate the following detailed explanations of the underlying concepts of turbo decoder and the iterative decoding performed by it, we first present the turbo decoder in its original form, which has been introduced in [2], in Figure 4. At this moment, we briefly outline the operation of the turbo decoder. Then, we will elaborate those concepts throughout the rest of this section.

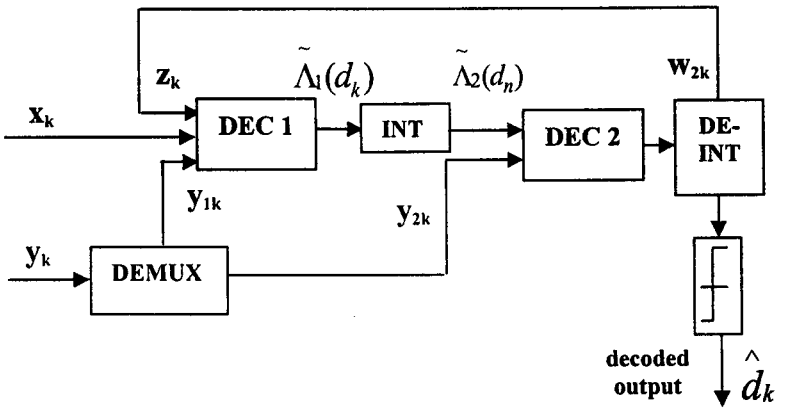

Figure 4. Turbo decoder

For the first iteration of the turbo decoder, z_k's are initialized to zero. Received bits x_k's and y_k's are demultiplexed to get the appropriate parity bits for the component decoders. Parity bits are provided to the corresponding decoders according to the puncturing pattern used in the turbo encoder. If the turbo code is unpunctured, then at each time index k y_{1k} and y_{2k} are provided to Decoder1 and Decoder2 respectively. If punctured turbo code is used and if one of the parity bits is deleted at every other time index k during the encoding process, then at that time index k the deleted parity bit is considered as "zero" for the corresponding decoder. For instance, if y_{1k} was deleted at time k during the encoding, then it is taken as "zero" for Decoder1, while y_{2k} is being used by Decoder2 at time k.

Decoder1 uses x_k and y_{1k} and produces a "soft" estimate of the decoded bit d_k. A relevant piece of information is extracted from this soft estimate (which is called as "extrinsic information") and passed on to Decoder2 after proper interleaving (same interleaver which is used in encoder). Decoder2 uses this extrinsic information along with x_k and y_{2k}, in turn yielding a better estimate of bit d_k. Then the extrinsic information produced by Decoder2 is passed on to Decoder1 for the next iteration to improve the estimates on all d_k's in the block. This iterative

process continues until a satisfactory BER is achieved. Usually, less than 20 iterations yield a BER of 10^{-5}, i.e. one bit in error out of onehundredthousand.

After a predetermined number of iterations is reached, soft output of Decoder2 is passed on to a hard limiter (with a threshold set to zero) to obtain the estimates on the information bits.

4. MODIFIED BCJR ALGORITHM

As we mentioned before, BCJR algorithm minimizes bit error probability in decoding linear block and convolutional codes, and yields the APP (a posteriori probability) for each decoded bit. In its original version given in [6], BCJR algorithm was shown to be directly applicable to decoding of convolutional codes. For decoding of RSC codes BCJR algorithm has to be modified in order to take into account the recursive nature of the encoder and that modified version of BCJR algorithm was given in [1] and [2]. However, we must emphasize that those modifications only stem from the fact that the encoder is recursive, and therefore have no significant effect on the essence of the algorithm, i.e. *"modified BCJR algorithm"* is also *"optimal"* in the sense that it minimizes the bit error probability as well. Since the modified version is in essence the same as the original algorithm in terms of the calculations of the APP probabilities and the operation of the algorithm, and since it is this modified version that is used in turbo decoders, we will directly present it without first introducing the original BCJR algorithm.

Let us consider an RSC code with constraint length K, memory v=K-1 (as given in Figure 2). At time k the encoder state S_k is respresented by a K-tuple:

$$S_k=(a_k, a_{k-1},\ldots\ldots, a_{k-K+2}) \tag{3}$$

Let us also consider that the information bit sequence $\{d_k\}$ consists of N independent bits d_k taking values zero and one with equal probability and that the encoder initial state S_0 and final state S_N are both equal to zero, i.e.

$$S_0 = S_N = (0,0,\ldots\ldots, 0) = \mathbf{0} \tag{4}$$

Let us also suppose that BPSK modulation is used, i.e. bit one is mapped to +1, and bit zero is mapped to −1. Note that here we assume that the average transmitted signal energy per information bit is normalized to one, i.e. $E_b = 1$.

The encoder output sequence is denoted by $C_1^N = \{C_1 \ldots\ldots C_k \ldots\ldots C_N\}$ where $C_k = (X_k, Y_k)$ is the input to a discrete Gaussian memoryless channel (DMC)

whose output is the sequence $R_1^N = \{R_1 ..... R_k ..... R_N\}$ where $R_k = (x_k, y_k)$ is defined by

$$\begin{aligned} x_k &= X_k + i_k \\ y_k &= Y_k + q_k \end{aligned} \tag{5}$$

where i_k and q_k are two independent noises with the same variance σ^2. As it can be seen from Figure 4, there is another variable, z_k, which is also fed to the decoder along with the received bits. As we will explain in detail later, z_k is the *"extrinsic information"* provided by the other component decoder to improve LLR (Logarithm of Likelihood Ratio) associated with each decoded bit d_k produced by the current decoder. In [1] and [2], z_k is approximated by another Gaussian variable with a variance $\sigma_z^2 \neq \sigma^2$, and it is weakly correlated with x_k and y_k. Note that in the following, we drop the sub-subscript, and instead of y_{1k} and y_{2k} we use y_k for simplicity. However, it should be kept in mind that y_k corresponds either y_{1k} or y_{2k} depending on which component decoder is under consideration.

"The logarithm of likelihood ratio (LLR)" associated with each decoded bit d_k is defined as:

$$\Lambda(d_k) = \ln \frac{\Pr\{d_k = 1 \mid observation)}{\Pr\{d_k = -1 \mid observation)} \tag{6}$$

$$\Lambda(d_k) = \ln \frac{\Pr\{d_k = 1 \mid R_1^N)}{\Pr\{d_k = -1 \mid R_1^N)} \tag{7}$$

Note that $\Pr\{d_k=i \mid \text{observation}\}$, i=-1,+1, is the *a posteriori probability* (APP) of the bit d_k. This APP can be derived from the joint probability $\lambda_k^i(m)$ defined by:

$$\lambda_k^i(m) = \Pr\{d_k = i, S_k = m \mid R_1^N\} \tag{8}$$

Using this, the APP of a decoded bit d_k can be written as:

$$\Pr\{d_k = i \mid R_1^N\} = \sum_m \lambda_k^i(m), \qquad i = -1,+1. \tag{9}$$

Then, from (7) and (9) the LLR $\Lambda(d_k)$ associated with each bit can be written as:

$$\Lambda(d_k) = \ln \frac{\sum_m \lambda_k^1(m)}{\sum_m \lambda_k^{-1}(m)} \tag{10}$$

If a hard decision is to be performed (or the decision at the end of an iterative process, as will be explained later), $\Lambda(d_k)$ can be compared to a threshold equal to zero to make a decision about the transmitted bit d_k:

$$\begin{aligned} \hat{d}_k &= +1 \quad if \quad \Lambda(d_k) \geq 0 \\ \hat{d}_k &= -1 \quad if \quad \Lambda(d_k) < 0 \end{aligned} \tag{11}$$

It is possible to rewrite the LLR $\Lambda(d_k)$ as follows:

$$\Lambda(d_k) = \ln \frac{\sum_m \sum_{m'} \Pr\{d_k = +1, S_k = m, S_{k-1} = m', R_1^{k-1}, R_k, R_{k+1}^N\}}{\sum_m \sum_{m'} \Pr\{d_k = -1, S_k = m, S_{k-1} = m', R_1^{k-1}, R_k, R_{k+1}^N\}} \tag{12}$$

Let us first prove that the joint probabilities which appeared both in the numerator and denominator of LLR expression given above can be written as the product of some probability functions which can be recursively calculated as will be shown later.

$$\begin{aligned} &\Pr\{d_k = i, S_k = m, S_{k-1} = m', R_1^N\} = \Pr\{d_k = i, S_k = m, S_{k-1} = m', R_1^{k-1}, R_k, R_{k+1}^N\} \\ &= \Pr\{R_{k+1}^N \mid d_k = i, S_k = m, S_{k-1} = m', R_1^{k-1}, R_k\} \\ &\Pr\{d_k = i, S_k = m, S_{k-1} = m', R_1^{k-1}, R_k\} \\ &= \Pr\{R_{k+1}^N \mid S_k = m\} \Pr\{d_k = i, S_k = m, R_k \mid S_{k-1} = m', R_1^{k-1}\} \Pr\{S_{k-1} = m', R_1^{k-1}\} \\ &= \Pr\{R_{k+1}^N \mid S_k = m\} \Pr\{d_k = i, S_k = m, R_k \mid S_{k-1} = m'\} \Pr\{S_{k-1} = m' \mid R_1^{k-1}\} \Pr\{R_1^{k-1}\} \end{aligned} \tag{13}$$

In the above derivation, we used the Bayes' rule successively, and took into account the fact that events after time k are not influenced by observation R_1^k and bit d_k if

the encoder state at time k, S_k, is known. This is due to the Markov property of the source. In the last step, $\Pr\{R_1^{k-1}\}$ is a constant and can be ignored because it will also appear in the denominator of LLR equation, and therefore will cancel out.

As given in [2], let us introduce some probability functions defined by

$$\alpha_k(m) = \Pr\{S_k = m \mid R_1^k\} \tag{14a}$$

$$\beta_k(m) = \frac{\Pr\{R_{k+1}^N \mid S_k = m\}}{\Pr\{R_{k+1}^N \mid R_1^k\}} \tag{14b}$$

$$\gamma_i(R_k, m', m) = \Pr\{d_k = i, S_k = m, R_k \mid S_{k-1} = m'\} \tag{14c}$$

Using these definitions we can rewrite LLR, $\Lambda(d_k)$, as:

$$\Lambda(d_k) = \ln \frac{\sum_m \sum_{m'} \gamma_1(R_k, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_{-1}(R_k, m', m)\alpha_{k-1}(m')\beta_k(m)} \tag{15}$$

It has been given in [2] that $\alpha_k(m)$ and $\beta_k(m)$ can be recursively calculated as:

$$\alpha_k(m) = \frac{\sum_{m'} \sum_{i=-1}^{+1} \gamma_i(R_k, m', m)\alpha_{k-1}(m')}{\sum_m \sum_{m'} \sum_{i=-1}^{+1} \gamma_i(R_k, m', m)\alpha_{k-1}(m')} \tag{16}$$

$$\beta_k(m) = \frac{\sum_{m'} \sum_{i=-1}^{+1} \gamma_i(R_{k+1}, m, m')\beta_{k+1}(m')}{\sum_m \sum_{m'} \sum_{i=-1}^{+1} \gamma_i(R_{k+1}, m, m')\alpha_k(m)} \tag{17}$$

Definition of $\alpha_k(m)$ given in [2] is slightly different than the one given in [1].

Now we can give the expression to evaluate $\gamma_i(R_k, m', m)$ which can be determined from transition probabilities of the discrete Gaussian memoryless

channel and transition probabilities of the encoder trellis. In [2], $\gamma_i(R_k, m', m)$ is written as the product of three probabilities:

$$\begin{aligned}\gamma_i(R_k, m', m) = p(R_k \mid d_k = i, S_{k-1} = m', S_k = m) \\ .q(d_k = i \mid S_{k-1} = m', S_k = m) \\ .\pi(S_k = m \mid S_{k-1} = m')\end{aligned} \tag{18}$$

where p(.|.) is the transition probability of the discrete Gaussian memoryless channel. Conditionally to $(d_k = i, S_{k-1} = m', S_k = m)$, x_k and y_k (R_k=(x_k,y_k,z_k)) are

two uncorrelated Gaussian variables and they are weakly correlated with z_k (due to interleaving), so we can write:

$$\begin{aligned}p(R_k \mid d_k = i, S_{k-1} = m', S_k = m) = p(x_k \mid d_k = i, S_{k-1} = m', S_k = m) \\ .p(y_k \mid d_k = i, S_{k-1} = m', S_k = m) \\ .p(z_k \mid d_k = i, S_{k-1} = m', S_k = m)\end{aligned} \tag{19}$$

Since the convolutional encoder is a deterministic machine, $q(d_k = i \mid S_{k-1} = m', S_k = m)$ is either 0 or 1, and can be determined from the encoder trellis. $\pi(S_k = m \mid S_{k-1} = m')$ are the transition state probabilities of the trellis, and are defined by the encoder input statistic. Generally, it is assumed that both input symbols are equally likely, i.e. $\Pr\{d_k=1\} = \Pr\{d_k=-1\}=1/2$ and as a result of this assumption $\pi(S_k = m \mid S_{k-1} = m') = 1/2$ since there are two possible transitions from each state. However, as we will explain shortly, in one of the two approaches $\pi(S_k = m \mid S_{k-1} = m')$ are considered as *"a priori probabilities"* and updated by each decoder using the *"extrinsic information"* provided by the other decoder in an iterative decoding scheme. Before we delve into those issues, let us summarize the Modified BCJR algorithm.

4.1. Summary of the Modified BCJR Algorithm

Step 1) Initialization of $\alpha_k(m)$ and $\beta_k(m)$ (from (4))

$$\begin{aligned}&\alpha_0(0) = 1 \\ &\alpha_0(m) = 0, \quad \forall m \neq 0\end{aligned} \tag{20a}$$

$$\begin{aligned}&\beta_N(0) = 1, \\ &\beta_N(m) = 0, \quad \forall m \neq 0\end{aligned} \tag{20b}$$

Step 2) For each observation R_k, $\alpha_k(m)$ is computed using (16) recursively and $\gamma_i(R_k, m', m)$ are computed using (18).

Step 3) When whole sequence R_1^N is received, $\beta_k(m)$ are computed using (17) recursively.

Step 4) LLR associated with each decoded bit d_k is computed using (15).

5. EXTRINSIC INFORMATION

In this section, we present one of the key concepts of turbo decoding, the so-called *"extrinsic information"*. It has been shown in [1] and [2] that the LLR, $\Lambda(d_k)$, associated with each decoded bit d_k can be written as the summation of the LLR of the d_k at the decoder input and extrinsic information generated by the decoder. Let us remember the definition of LLR given in (15) and also remember that $R_k = (x_k, y_k, z_k)$ where x_k is the received noisy information bit, y_k is the received noisy parity bit and z_k is the extrinsic information provided by the other component decoder.

As given in (19), $\gamma_i(x_k, y_k, z_k, m', m)$ can be written as the product of three conditional probabilities. Also, since the encoder is systematic ($X_k = d_k$), the transition probabilities $p(x_k \mid d_k = i, S_k = m, S_{k-1} = m')$ and $p(z_k \mid d_k = i, S_k = m, S_{k-1} = m')$ are independent of the state transitions. Therefore $\Lambda(d_k)$ can be written as:

$$\Lambda(d_k) = \ln \frac{p(x_k \mid d_k = 1)}{p(x_k \mid d_k = -1)} + \ln \frac{p(z_k \mid d_k = 1)}{p(z_k \mid d_k = -1)} + \ln \frac{\sum_m \sum_{m'} \gamma_1(y_k, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_{-1}(y_k, m', m)\alpha_{k-1}(m')\beta_k(m)} \tag{21}$$

Variables x_k are Gaussian with mean 1 and –1 and variance σ^2, conditionally to $d_k = 1$ and $d_k = -1$ respectively. Hence, $p(x_k \mid d_k = i)$ can be written as:

$$p(x_k \mid d_k = i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\frac{-(x_k - i)^2}{2\sigma^2}\right] \tag{22}$$

Therefore, we can write the first term in (21) as:

$$\ln\frac{p(x_k \mid d_k=1)}{p(x_k \mid d_k=-1)} = \ln p(x_k \mid d_k=1) - \ln p(x_k \mid d_k=-1)$$

$$\ln\frac{p(x_k \mid d_k=1)}{p(x_k \mid d_k=-1)} = \ln\{\frac{1}{\sqrt{2\pi\sigma^2}}\exp[\frac{-(x_k-1)^2}{2\sigma^2}]\} - \ln\frac{1}{\sqrt{2\pi\sigma^2}}\exp[\frac{-(x_k+1)^2}{2\sigma^2}]\}$$

$$\ln\frac{p(x_k \mid d_k=1)}{p(x_k \mid d_k=-1)} = \ln(\frac{1}{\sqrt{2\pi\sigma^2}}) + [\frac{-(x_k-1)^2}{2\sigma^2}] - \ln(\frac{1}{\sqrt{2\pi\sigma^2}}) - [\frac{-(x_k+1)^2}{2\sigma^2}]$$

$$\ln\frac{p(x_k \mid d_k=1)}{p(x_k \mid d_k=-1)} = \frac{-x_k^2+2x_k-1+x_k^2+2x_k+1}{2\sigma^2} = \frac{2}{\sigma^2}x_k \tag{23}$$

In [1] and [2], it has been shown that z_k can be approximated by another Gaussian variable with variance σ_z^2 and as has been mentioned this Gaussian assumption, even if it is not satisfied for the first few iterations, becomes a good approximation as the number of the iterations increases. With this approach, variance σ_z^2 and the mean value of this Gaussian variable has to be estimated at each iteration as explained in [1] and [2]. From (21) and (23), we can finally write $\Lambda(d_k)$ as:

$$\Lambda(d_k) = \frac{2}{\sigma^2}x_k + \frac{2}{\sigma_z^2}z_k + w_k \tag{24}$$

where

$$w_k = \ln\frac{\sum_m\sum_{m'}\gamma_1(y_k,m',m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m\sum_{m'}\gamma_{-1}(y_k,m',m)\alpha_{k-1}(m')\beta_k(m)} \tag{25}$$

is defined as the **"extrinsic information"** in [1] and [2]. Note that w_k is a function of the parity bit y_k, or the redundant information introduced by the corresponding component encoder. Therefore, it does not depend on the decoder input x_k and the extrinsic information z_k provided by the other component decoder in the feedback scheme. As it has been pointed out in [1] and [2], the extrinsic information has

proven to be a useful piece of information which is used by the other decoder to perform a new decoding for a better estimate of the information bit d_k. This iterative procedure is repeated until a desired BER (bit error rate) is achieved. For a punctured turbo code of rate R=1/2 encoder, with memory v=4 and code generators G_1=37, G_2=21, a BER of 10^{-5} is achieved by 18 iterations in [1] and [2].

We should emphasize that z_k must not be used by the next decoder, because it was produced by it in the previous iteration. Therefore, next decoder must use the extrinsic information w_k in addition to the scaled systematic information bit $\frac{2}{\sigma^2}x_k$.

5.1 Extrinsic Information Used to Update the A Priori Probabilities

In this section, we describe a more efficient use of extrinsic information. Colavolpe et al. in a paper entitled as "Extrinsic Information in Iterative Decoding: A Unified View" [5] compared the two different approaches regarding the use of the extrinsic information. First approach is based on the use of extrinsic information as another Gaussian variable, as we explained in a previous section. Their work showed that the other approach, which we will explain in detail in the following, while outperforming the Gaussian approximation approach in terms of BER, does not require estimation of the mean value and the variance of z_k at each iteration, when z_k is considered as a Gaussian random variable.

In this approach extrinsic information is used to update the *"a priori probabilities"* of the input information bits, namely $\Pr\{d_k=1\}$ and $\Pr\{d_k=-1\}$. Earlier we mentioned that input bits +1 and –1 are assumed to be equally likely, i.e. $\Pr\{d_k=1\}=\Pr\{d_k=-1\}=1/2$ and as a result of this assumption $\pi(S_k = m / S_{k-1} = m') = 1/2$ since there are two possible transitions from each state. However, extrinsic information z_k provided by the other component decoder can be used to update this "a priori probabilities" as follows.

$$\begin{aligned} &\text{If } q(d_k = 1 \mid S_k = m, S_{k-1} = m') = 1, \text{ then} \\ &\qquad \Pr\{d_k = 1\} = \pi(S_k = m \mid S_{k-1} = m') \\ &\qquad \Pr\{d_k = -1\} = 1 - \pi(S_k = m \mid S_{k-1} = m') \end{aligned} \tag{26}$$

$$\begin{aligned} &\text{If } q(d_k = -1 \mid S_k = m, S_{k-1} = m') = 1, \text{ then} \\ &\qquad \Pr\{d_k = 1\} = 1 - \pi(S_k = m \mid S_{k-1} = m') \\ &\qquad \Pr\{d_k = -1\} = \pi(S_k = m \mid S_{k-1} = m') \end{aligned} \tag{27}$$

When it is used as a priori information z_k can be written as:

$$z_k = \ln \frac{\Pr\{d_k = 1\}}{\Pr\{d_k = -1\}} \tag{28}$$

Let us suppose that $q(d_k = 1 \mid S_k = m, S_{k-1} = m') = 1$, and then we can write z_k as:

$$z_k = \ln \frac{\Pr\{d_k = 1\}}{\Pr\{d_k = -1\}} = \ln \frac{\pi(S_k = m \mid S_{k-1} = m')}{1 - \pi(S_k = m \mid S_{k-1} = m')} \tag{29}$$

$$\exp(z_k) = \frac{\pi(S_k = m \mid S_{k-1} = m')}{1 - \pi(S_k = m \mid S_{k-1} = m')} \tag{30}$$

Similarly, we can also find $\exp(z_k)$ when $q(d_k = -1 \mid S_k = m, S_{k-1} = m') = 1$.

Finally, we can obtain $\pi(S_k = m \mid S_{k-1} = m')$ as:

$$\pi(S_k = m \mid S_{k-1} = m') = \left\{ \begin{array}{ll} \dfrac{\exp(z_k)}{1 + \exp(z_k)} & \text{if } q(d_k = 1 \mid S_k = m, S_{k-1} = m') = 1 \\ \dfrac{1}{1 + \exp(z_k)} & \text{if } q(d_k = -1 \mid S_k = m, S_{k-1} = m') = 1 \end{array} \right\} \tag{31}$$

This can be written as:

$$\pi(S_k = m \mid S_{k-1} = m') =$$

$$\left\{ \begin{array}{ll} \dfrac{\exp(z_k / 2)}{1 + \exp(z_k)} \exp(z_k / 2) & \text{if } q(d_k = 1 \mid S_k = m, S_{k-1} = m') = 1 \\ \dfrac{\exp(z_k / 2)}{1 + \exp(z_k)} \exp(-z_k / 2) & \text{if } q(d_k = -1 \mid S_k = m, S_{k-1} = m') = 1 \end{array} \right\}$$

$$\pi(S_k = m \mid S_{k-1} = m') = \frac{\exp(z_k / 2)}{1 + \exp(z_k)} \exp(d_k z_k / 2) \tag{32}$$

In this case, LLR associated with each decoded bit d_k, $\Lambda(d_k)$, is expressed as:

$$\Lambda(d_k) = z_k + \ln \frac{\sum_m \sum_{m'} \gamma_1(x_k, y_k, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_{-1}(x_k, y_k, m', m)\alpha_{k-1}(m')\beta_k(m)} \tag{33}$$

We should keep in mind that z_k must not be used by the next decoder since it was produced by it in the previous iteration.

6. CALCULATION OF $\gamma_i(R_k, m', m)$

Now, we have everything ready to calculate the $\gamma_i(R_k, m', m)$. We should emphasize that now $R_k = (x_k, y_k)$, because z_k is used to update the a priori probabilities as explained above. From (18) and (19) we can write $\gamma_i(R_k, m', m)$ as:

$$\begin{aligned} \gamma_i(R_k, m', m) &= p(x_k \mid d_k = i) \\ &\quad .p(y_k \mid d_k = i, S_k = m, S_{k-1} = m') \\ &\quad .q(d_k = i \mid S_k = m, S_{k-1} = m') \\ &\quad .\pi(S_k = m \mid S_{k-1} = m') \end{aligned} \tag{34}$$

Using (22) and (32):

$$\begin{aligned} \gamma_i(R_k, m', m) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\frac{-(x_k - i)^2}{2\sigma^2}\right] \\ &\quad . \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\frac{-(y_k - Y_k)^2}{2\sigma^2}\right] \\ &\quad .q(d_k = i \mid S_{k-1} = m', S_k = m) \\ &\quad . \frac{\exp(z_k/2)}{1 + \exp(z_k)} \exp(iz_k/2) \end{aligned} \tag{35}$$

$$\begin{aligned} \gamma_i(R_k, m', m) &= \frac{1}{2\pi\sigma^2} \frac{\exp(z_k/2)}{1 + \exp(z_k)} \exp(iz_k/2) \\ &\quad .q(d_k = i \mid S_k = m, S_{k-1} = m') \\ &\quad .\exp\left\{\frac{-1}{2\sigma^2}\left[(x_k - i)^2 + (y_k - Y_k)^2\right]\right\} \end{aligned} \tag{36}$$

where Y_k is the parity bit generated by the encoder when $d_k = i, S_{k-1} = m', S_k = m$, and it is known from the encoder trellis, being either –1 or +1. And as we mentioned

earlier, $q(d_k = i \mid S_{k-1} = m', S_k = m)$ is either 0 or 1, and can also be determined from the encoder trellis.

7. CONCLUSIONS

In this paper, we presented a thorough review of turbo codes and their iterative decoding technique, so-called turbo decoding, and the underlying concepts. Also, we elaborated on the BCJR (Bahl algorithm), the most widely used algorithm in turbo decoding.

ÖZET

Turbo kodlama ve kod çözme, kanal kodlama alanında yeni bir dönem açmıştır. Bu makalede, tüm bağıntıları ayrıntılı bir şekilde türetilerek turbo kodlama ve kod çözmenin bir incelemesi yapılmıştır. Ayrıca, kod çözmede en yaygın kullanılan BCJR (Bahl, Cocke, Jelinek and Raviv) algoritması (aynı zamanda maksimum a posteriori algoritması – MAP algoritması olarak da bilinen) da ayrıntılı olarak verilmiştir.

REFERENCES

[1] C. Berrou, A. Glavieux, P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. *International Conference on Communications, ICC'93*, 2:1064-1070, 23-26 May 1993.

[2] C. Berrou, A. Glavieux. Near optimum error-correcting coding and decoding: Turbo-codes. *IEEE Transactions on Communications*, 44:10:1261-1271, 1996.

[3] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379-423, 623-656, October 1948.

[4] C. Heegard, S. B. Wicker. Turbo Coding. *Kluwer Academic Publishers*, 1999.

[5] G. Colavolpe, G. Ferrari, R. Raheli. Extrinsic information in iterative decoding: A unified view. *IEEE Transactions on Communications*, 49:12:2088-2094, 2001.

[6] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, 20:2:284-287, March 1974.

[7] M. H. Sazli, Neural Network Applications to Turbo Decoding. Ph.D. Dissertation, Syracuse University, 2003.

C O M M U N I C A T I O N S

DE LA FACULTE DES SCIENCES
DE L'UNIVERSITE D'ANKARA

FACULTY OF SCIENCES
UNIVERSITY OF ANKARA

Séries A2-A3: Physics, Engineering Physics and Astronomy

INSTRUCTIONS TO CONTRIBUTORS

Communications accepts original research articles and letters to the Editor in various fields of research in physics, engineering physics and astronomy. Contribution is open to researchers of all nationalities.

Manuscripts should be written in English. Each paper should be preceded by an abstract in English. They must be type-written using a font style and size which is quite legible, double-spaced throughout with ample margins and single-sided on white A4 standard paper. Manuscripts submitted for publication should contain one original and two complete copies; single or incomplete texts will not be accepted and will not be returned.

After the manuscript has been accepted for publication, i.e., after referee-recommended revisions are complete, the author will not be permitted to make any new additions to the manuscript.

Attention: before publication the galley proof is always sent to the author for corrections. Thus, it is solely the author's responsibility for any typographical mistakes which occur in their article as it appears in the journal. Only those mistakes/omissions which occur due to some negligence on our part during the final printing will be reprinted in a Corrections section of a later issue. However, this does not include those errors left unnoticed by the author on the galley proofs.

1. Title Page

The title should be short in length but informative. Each title page should contain:

(i) Name of the paper, (ii) Complete name(s) of the author(s), (iii) Name and address of the university, laboratory or institute at which the reseacrh has been carried out.

2. Abstract

Each paper should always be preceded by an abstract. The abstract should be short and self-contained and all essential points of the paper should be mentioned.

3. Sections and Subsections

Principle sections such as introduction or formulation of the problem should be numbered consecutively (1. Introduction, 2. Formulation of the problem, ..., etc.) Subsections should be numbered 1.1, 1.2, ..., etc.

4. References

References including comments must be numbered consecutively in order of first appearance in the text. The reference number should be put in brackets [1] where referred to in the text. References should be listed at the end of the manuscript in the numbered order in which they appear in the text. The method of citation should be as follows:

Articles in Periodicals: (i) Surname(s) of the author(s) with their initial(s), (ii) Title of the periodical abbreviated according to the "Physics Abstract list of journals", (iii) Volume number, (iv) Year of publication, (v) Page number.

Example: M. Chen and P.M. Zerwas, Phys. Rev. D 12 (1975) 187.

Books; (i) Surname(s) of the author(s) with their initial(s), (ii) Title of the book, (iii) Name of the editor (if any), (iv) Volume number, (v) Place and year of publication and name of the publisher, (vi) Page number.

Example: A. Arima, Progress in particle and nuclear physics, ed. D. Wilkinson, vol. 1 (Pergamon, New York, 1978) p. 41.

Theses: (i) Surname and initials of the author, (ii) Type of degree (Ph.D., M.Sc.), (iii) Name and address of institute where the work has been carried out, (iv) Year.

Example: H. Parker, Ph D Thesis, Department of Physics, Univ. of California, Davis, CA 95616, USA, 1990.

5. Footnotes

Footnotes should be avoided if possible, but when necessary, should be short and never contain any important part of the work and should be numbered consecutively by superscripts.

6. Tables and Figures

All illustrations not including tables (photographs and other films, drawings, graphs, etc) must be labeled as "Figure".

The proper position of each table and figure must be clearly indicated in the paper.

All tables and figures must have a number (Table 1, Figure 1) and a caption or legend. If there is only one table or figure simply label it "Table" or "Figure".

All tables and figures must be numbered consecutively throughout the paper.

All captions and legends must also be double-space typed on a separate sheet and labeled according to which table or figure they belong.

Dimensions of tables and figures must not exceed 13x16 cm.

Tables must be clearly typed, each on a separate sheet, and double-spaced. Dimensions including caption, title, column heads, and footnotes must not exceed 13x16 cm. Tables may be continued onto another sheet if necessary but dimensions still apply.

Figures must be originals and drawn clearly in Indian ink on white paper or printed on smooth tracing/drawing paper. Reduced photocopies are not acceptable. Photographs must be clear, black and white, and printed on glossy paper.

7. Appendices

All appendices must be typed on separate sheets and should be numbered consecutively with capital Roman numerals.

8. Computer Disk:

If you are able to initially prepare your manuscript in a MS Word Programme (Macintosh or PC) file including the figures translated into the picture environment of Encapsulated PostScript format (EPS), we advise that you do so. Then, only if and after your manuscript is accepted for publication, we will ask you to submit a revised disk copy of your manuscript which will enable us to more efficiently and accurately prepare proofs. (This is not a requirement but is highly encouraged.)

9. Address:

Texts should be sent to the following address:
Prof.Dr. Öner ÇAKAR - Editor, Communications
Ankara Üniversitesi, Fen Fakültesi
06100, Beşevler-ANKARA

COMMUNICATIONS

DE LA FACULTE DES SCIENCES
DE L'UNIVERSITE D'ANKARA

FACULTY OF SCIENCES
UNIVERSITY OF ANKARA

Volume : 49 **Number: 1** **Year : 2005**

Series : A2-A3