Turbo Codes and Turbo Decoding

Kwame Ackah Bohulu

20-04-2017

1 Introduction

A typical Turbo encoder is formed fom two convolutional encoders, and an interleaver. The two convolutional encoders, also known as component encoders are connected in parallel via the interleaver. It was discovered by Claude Berrou and introduced to the scientific community in 1993[2]. In the usual case, two identical convolutional encoders are used. The order of the information sequence is rearranged via an interleaver before being fed into the second convolutional interleaver. The interleaver in the turbo encoder is used to reduce the number of low-weight codewords. The minimum achievable output rate of the interleaver is 1/3 but can be increased to 1/2 or 2/3 via puncturing. Turbo codes achieve excellent performance at low signal to noise ratios and have been used in many applications including mobile telephony standards. In this research paper, a brief introduction to turbo codes followed by the principle behind the construction of turbo codes are given in section 2 and 3 respectively. A detailed explanation of the BCJR algorithm and its application in the iterative decoding is given in section 4. Finally in section 5, the system model used in the simulation of the turbo code is explained in detail.

2 Convolution Codes

Convolutional codes are foremed by feeding an information sequence into finite state shift registers[1]. In general, the shift register consists of K stages each containing k bits and n linear algebraic function generators, where $K$is the constraint length of the shift register, $k$is the number of input bits fed into the shift register at a go,and n is the number of bits present at the output of the shift register. The rate of the convolutional encoder is $\frac{k}{n}$. The linear algebraic function generator perform modulo 2 addition. The generator function of the convolutional encoder is made up of n vector each of size Kk. Various representations of genetator matrix are shown below.

K=3,k=1,n=3
$g_1 = [100], g_2 = [101], g_3 = [111]$
octal form =(4, 5, 7)

$c^{(1)} = u * g_1, c^{(2)} = u * g_2, c^{(3)} = u * g_3$
Output code sequence=$(c_1^{(1)}, c_1^{(2)}, c_1^{(3)}, c_2^{(1)}, c_2^{(2)}, c_2^{(3)}, ...)$

D domain
$U(D) = \sum_{i=0}^{\infty} u_i D^i$
$g_1(D) = 1, g_2(D) = 1 + D^2, g_3(D) = 1 + D + D^2$
$C^{(1)}(D) = U(D)g_1(D), C^{(2)}(D) = U(D)g_2(D), C^{(3)}(D) = U(D)g_3(D)$
$C(D) = C^{(1)}(D^3) + DC^{(2)}(D^3) + D^2C^{(3)}(D^3)$

Convolutional codes are best described using a trellis graph. The number of nodes contained in the trellis is eqivalent to the number of possible states of the shift register. for a general rate$\frac{k}{n}$ code with constraint length K , the trellis has $2^{k(K-1)}$ossible nodes with 2^k branches entering and leaving each node.

2.0.1 Recursive Systematic and Nonrecursive Systematic Convolutional Codes

Convolutional codes which have the information sequence as part of the code are known as systematic convolutional codes. Futhermore, a convolutional code constructed by the use of a feedback shift register is known as a recursive convolutional code. It should be noted that recursive convolutional code are always sytematic whiles nonrecursive convolutional codes are often non-systematic. Also, the generator matrix for systematic convolutional codes contain a polynomial matrix.

rate=1/2, K=3, $g_1 = [111]$, $g_2 = [101]$
G=[1 g_2/g_1]

The "1" in the generator matrix signifies the systematic portion of the code whiles denominator and numerator signifies the feedback input and the output respectively. Recursive systematic encoders are used as component codes in most Turbo code applications.

3 Turbo Encoder

As was mentioned earlier, a turbo encoder is made up of two recursive systematic convolutional(RSC) encoders connected in parallel by an interleaver[4]. The choice of interleaver is very important, as it caused a reduction in the number of low-weight codewords(called the multiplicity)[3],[5]. Due to the huge number of states that exist in the turbo encoder, maximum likelihood decoding is impossible. Instead, the turbo decoding algorithm, which is based on the BCJR algorithm is used.

4 BCJR Algorithm

The BCJR algorithm was invented by Bahl, Cocke,Jelinek and Raviv in 1974. Rather than searching for the most likely input sequence, this algorithm uses the MAP algorithm to decode each input sequence. in order to explain the BCJR algorithm, the following assumptions are made
1. The information sequence length, **u**is N,**u**=$\{u_1, u_2, ..., u_N\}$
In the case where ,$k = 1$, $u_i \in \{0, 1\}$
2. The encoded sequence length **c**is N,**c**=$\{c_1, c_2, ..., c_N\}$
and the length ofc_i is n.
3.The encoded sequence is transmitted through an AWGN channel and the received sequence **y**is real and has length n.**y**=$\{y_1, y_2, ..., y_N\}$

The BCJR algorithm calculates the a posteriori LLR,$L(u_i|y)$ and returns an estimate of the original information sequence using the equation below.

$$L(u_i|y) = \ln \frac{P(u_i = 1|\boldsymbol{y})}{P(u_i = 0|\boldsymbol{y})} \tag{1}$$

The calculation of the BCJR algorithm is made easier by the use of a trellis graph。At time i, the current state is given as $\sigma_i = s$ and the previous state is given by $\sigma_{i-1} = s'$. The received symbol at the decoder at time i, is given by y_i. Before time i,$i-1$ symbols would have been received and after time I $N-i$ symbols will be received. From the above intuition, the total received sequence $\mathbf{y}$ at time i can be divided in past present and future subsequences、

$$\boldsymbol{y} = \boldsymbol{y}_{<i}\boldsymbol{y}_i\boldsymbol{y}_{>i}$$

From the trellis graph, we note that the transition from $\sigma_{i-1} = s'$ to $\sigma_i = s$ is determined by the value of u_i .Since state transitions are mutually exclusive, the probability that a transition will occur is the sum of the respective probabilities.

$$\therefore P(u_i = 1|\boldsymbol{y}) = \sum_{R_1} P(s', s|\boldsymbol{y})$$

$$P(u_i = 0|\boldsymbol{y}) = \sum_{R_0} P(s', s|\boldsymbol{y})$$

where
R_0 is the set transition from $\sigma_{i-1} = s'$ to σ_i=s as a result of $u_i = 0$
R_1 is the set transition from $\sigma_{i-1} = s'$ to σ_i=s as a result of $u_i = 1$

substituting $\mathbf{y}$,$P(u_i = 1|\boldsymbol{y})$ and $P(u_i = 0|\boldsymbol{y})$ into equation 1、

$$\begin{aligned} L(u_i|y) &= \ln \frac{\sum_{R_1} P(s', s|\boldsymbol{y})}{\sum_{R_0} P(s', s|\boldsymbol{y})} \\ &= \ln \frac{\sum_{R_1} P(s', s, \boldsymbol{y})}{\sum_{R_0} P(s', s, \boldsymbol{y})} \\ &= \ln \frac{\sum_{R_1} P(s', s, \boldsymbol{y}_{<i}\boldsymbol{y}_i\boldsymbol{y}_{>i})}{\sum_{R_0} P(s', s, \boldsymbol{y}_{<i}\boldsymbol{y}_i\boldsymbol{y}_{>i})} \end{aligned} \tag{2}$$

from Bayes equation

$$\begin{aligned} P(s', s, \boldsymbol{y}_{<i}\boldsymbol{y}_i\boldsymbol{y}_{>i}) &= P(\boldsymbol{y}_{>i}|s', s, \boldsymbol{y}_{<i}\boldsymbol{y}_i)P(s', s, \boldsymbol{y}_{<i}\boldsymbol{y}_i) \\ &= P(\boldsymbol{y}_{>i}|s)P(\boldsymbol{y}_i, s|s', \boldsymbol{y}_{<i})P(s', \boldsymbol{y}_{<i}) \\ &= P(\boldsymbol{y}_{>i}|s)P(\boldsymbol{y}_i, s|s')P(s', \boldsymbol{y}_{<i}) \\ &= \alpha_{i-1}(s')\gamma_i(s', s)\beta_i(s) \end{aligned} \tag{3}$$

$\alpha_{i-1}(s') = P(s', \boldsymbol{y}_{<i})$ represent the joint probability that at time $i-1$、the state s' and the received sequence untill then is $\boldsymbol{y}_{<i}$.
$\gamma_i(s', s) = P(\boldsymbol{y}_i, s|s')$ represents the probability that the next state is s and the received symbol is $\boldsymbol{y}_i$ given that the previous state is s.
$\beta_i(s) = P(\boldsymbol{y}_{>i}|s)$ represents the conditional probability tha given the current state, s, the future sequence will be $\boldsymbol{y}_{>i}$ Substituting equation 3 into equation 1 yields the equation below.

$$L(u_i|y) = \ln \frac{\sum_{R_1} \alpha_{i-1}(s')\gamma_i(s', s)\beta_i(s)}{\sum_{R_0} \alpha_{i-1}(s')\gamma_i(s', s)\beta_i(s)} \tag{4}$$

4.1 Calculation of $\gamma_i(s', s)$

$\gamma_i(s', s)$ is given by the equation below.

$$\begin{aligned} \gamma_i(s', s) &= P(\boldsymbol{y}_i, s|s') \\ &= P(\boldsymbol{y}_i|s', s)P(s|s') \\ &= P(\boldsymbol{y}_i|c_i)P(u_i) \end{aligned} \tag{5}$$

In the case of the AWGN channel、

$$\gamma_i(s', s) = \frac{P(u_i)}{(\pi N_o)^{n/2}} exp(-\frac{\|y_i - c_i\|^2}{N_o}) \tag{6}$$

4.2 Calculation of $\alpha_{i-1}(s')$

$\alpha_{i-1}(s') = P(s', \boldsymbol{y}_{<i})$. This equation can be re-written as、

$$\begin{aligned} \alpha_i(s) &= P(s, \boldsymbol{y}_{<i+1}) \\ &= P(s, \boldsymbol{y}_{<i}, \boldsymbol{y}_i) \end{aligned} \tag{7}$$

From probability theory,

$$P(A) = \sum_{B} P(A, B) \tag{8}$$

$$\begin{aligned} \therefore \alpha_i(s) &= \sum_{s'} P(s, \boldsymbol{y}_i | s', \boldsymbol{y}_{<i}) P(s', \boldsymbol{y}_{<i}) \\ &= \sum_{s'} P(s, \boldsymbol{y}_i | s') P(s', \boldsymbol{y}_{<i}) \\ &= \gamma_i(s', s) \alpha_{i-1}(s') \end{aligned} \tag{9}$$

In the case of an all-zero terminated trellis、the initial conditions for $\alpha_i(s)$ are

$$\alpha_i(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases}$$

4.3 Calculation of $\beta_i(s)$

$\beta_i(s) = P(\boldsymbol{y}_{>i} | s)$ This can be re-written as

$$\begin{aligned} \beta_{i-1}(s') &= P(\boldsymbol{y}_{>i-1} | s') \\ &= \sum_{s} P(\boldsymbol{y}_{>i} | s', s, \boldsymbol{y}_i) P(s, \boldsymbol{y}_i | s') \\ &= \sum_{s} P(\boldsymbol{y}_{>i} | s) P(s, \boldsymbol{y}_i | s') \\ &= \beta_i(s) \gamma_i(s', s) \end{aligned} \tag{10}$$

In the case of an all-zero terminated trellis、the initial conditions for $\beta_i(s)$ are

$$\beta_N(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases}$$

4.4 Log-MAP and Max-Log-MAP

For the BCJR Algorithm、when the trellis length is very long、it becomes numerically unstable. In such cases, the log domain versions of the BCJR algorithm、Log-MAP andMax-Log-MAPAlgorithm are used instea. In the case of Log-MAP the following definitions are used.

$$\begin{aligned} \widetilde{\alpha}_i(s) &= \ln(\alpha(s)) \\ \widetilde{\beta}_i(s) &= \ln(\beta(s)) \\ \widetilde{\gamma}_i(s', s) &= \ln(\gamma(s', s)) \end{aligned} \tag{11}$$

The forward($\widetilde{\alpha}_i(s)$) and backward($\widetilde{\beta}_{i-1}(s)$) recursions as well as the LLR are calculated using the equations below.

$$\widetilde{\alpha}_i(s) = \ln \sum_{s'} \exp(\widetilde{\alpha}_{i-1}(s') + \widetilde{\gamma}(s', s))$$

$$\widetilde{\beta}_{i-1}(s) = \ln \sum_{s} \exp(\widetilde{\beta}_i(s') + \widetilde{\gamma}(s', s))$$

$$L(u_i) = \ln \Big[\sum_{R_1} \exp\ \widetilde{\alpha}_{i-1}(s') + \widetilde{\gamma}(s', s) + (\widetilde{\beta}_i(s')) \Big] - \ln \Big[\sum_{R_0} \exp(\widetilde{\alpha}_{i-1}(s') + \widetilde{\gamma}(s', s) + (\widetilde{\beta}_i(s')) \Big] \tag{12}$$

In order to increase calculation efficiency、Max-Log-MAP algorithm is used. The following definitions are used.

$$\begin{aligned} &\max *\{x, y\} \triangleq \ln(e^x + e^y) \\ &\max *\{x, y, z\} \triangleq \ln(e^x + e^y + e^z) \end{aligned} \tag{13}$$

With the above notations, The forward and backward recursions as well as the LLR are calculated using the equations below.

$$\widetilde{\alpha}_i(s) = \max_{s'} *\{\widetilde{\alpha}_{i-1}(s') + \widetilde{\gamma}(s', s)\}$$

$$\widetilde{\beta}_{i-1}(s) = \max_{s} *\{\widetilde{\beta}_i(s') + \widetilde{\gamma}(s', s)\}$$

$$L(u_i) = \max_{R_1} *\Big\{ \widetilde{\alpha}_{i-1}(s') + \widetilde{\gamma}(s', s) + (\widetilde{\beta}_i(s') \Big\} - \max_{R_0} *\Big\{ \widetilde{\alpha}_{i-1}(s') + \widetilde{\gamma}(s', s) + (\widetilde{\beta}_i(s') \Big\} \tag{14}$$

In the case of an all-zero terminated trellis, in initial conditions for $\widetilde{\alpha}_0$ and $\widetilde{\beta}_N(s)$ for both algorithms is given by

$$\widetilde{\alpha}_0(s) = \begin{cases} 0, & s = 0 \\ -\infty, & s \neq 0 \end{cases}$$

$$\widetilde{\beta}_N(s) = \begin{cases} 0, & s = 0 \\ -\infty, & s \neq 0 \end{cases}$$

5 Turbo Decoding Algorithm

Due to the large number of states available in the turbo encoder, maximum likelihood decoding is impossible. Instead、the Turbo decoding algorithm [4] proposed by Claude Berrou is used. This algorithm is based on the iterative use of the Log-MAP or Max-Log-MAP algorithm. For calculation efficiency this

research paper makes use of the Max-Log-MAP algorithm.For $n = 2$, AWGN channel and BPSK modulation、

$$\boldsymbol{c}_i = (c_i^s, c_i^p), \boldsymbol{y}_i = (y_i^s, y_i^p)$$

Equation 6 becomes

$$\gamma_i(s', s) = \frac{1}{(\pi N_o)} \exp(-\frac{(y_i^s)^2 + (y_i^p)^2 + 2(c_i^s)^2}{N_o}) P(u_i) \exp(\frac{2y_i^s c_i^s + 2y_i^p c_i^p}{N_o})$$

Since $\frac{1}{(\pi N_o)} \exp(-\frac{(y_i^s)^2+(y_i^p)^2+2(c_i^s)^2}{N_o})$ has no relation to u_i, it can be ignored. This yields the following equation.

$$\begin{aligned} \gamma_i(s', s) = P(u_i) \exp(\frac{2y_i^s c_i^s + 2y_i^p c_i^p}{N_o}) \\ P(u_i) \exp(\frac{2y_i^s c_i^s}{N_o}) \exp(\frac{2y_i^p c_i^p}{N_o}) \end{aligned} \tag{15}$$

$$\therefore \widetilde{\gamma}(s', s) = \ln P(u_i) + \exp(\frac{2y_i^s c_i^s}{N_o}) + \exp(\frac{2y_i^p c_i^p}{N_o})$$

Inserting $\widetilde{\gamma}(s', s)$into equation 14 $L(u_i)$ becomes,

$$\begin{aligned} L(u_i) = &\max_{R_1} * \Big\{ \widetilde{\alpha}_{i-1}(s') + [\ln P(u_i) + \exp(\frac{2y_i^s c_i^s}{N_o}) + \exp(\frac{2y_i^p c_i^p}{N_o})] + (\widetilde{\beta}_i(s') \Big\} - \\ &\max_{R_0} * \Big\{ \widetilde{\alpha}_{i-1}(s') + [\ln P(u_i) + \exp(\frac{2y_i^s c_i^s}{N_o}) + \exp(\frac{2y_i^p c_i^p}{N_o})] + (\widetilde{\beta}_i(s') \Big\} \end{aligned} \tag{16}$$

Assuming

$$c_i^s = \begin{cases} \sqrt{\varepsilon_c}, & u_i = 1 \\ -\sqrt{\varepsilon_c}, & u_i = 0 \end{cases}$$

yields,

$$\begin{aligned} L(u_i) = &\frac{4\sqrt{\varepsilon_c} y_i^s}{N_o} + \ln \frac{P(u_i) = 1}{P(u_i) = 0} + \max_{R_1} * \Big\{ \widetilde{\alpha}_{i-1}(s') + \exp(\frac{2y_i^p c_i^p}{N_o}) + (\widetilde{\beta}_i(s') \Big\} - \\ &\max_{R_0} * \Big\{ \widetilde{\alpha}_{i-1}(s') + \exp(\frac{2y_i^p c_i^p}{N_o}) + (\widetilde{\beta}_i(s') \Big\} \\ &= L_c y_i^s + L^{(a)}(u_i) + L^{(e)}(u_i) \end{aligned} \tag{17}$$

$L_c y_i^s = \frac{4\sqrt{\varepsilon_c} y_i^s}{N_o}$ is the channel $L(u_i)$ value、and denotes the effect of channel output corresponding to the systematic bits.

$L^{(a)}(u_i) = \ln \frac{P(u_i=1}{P(u_i)=0}$ is the a priori)$L(u_i)$ value.

$$L^{(e)}(u_i) = \max_{R_1} * \Big\{ \widetilde{\alpha}_{i-1}(s') + \exp(\frac{2y_i^p c_i^p}{N_o}) + (\widetilde{\beta}_i(s') \Big\} - \max_{R_0} * \Big\{ \widetilde{\alpha}_{i-1}(s') + \exp(\frac{2y_i^p c_i^p}{N_o}) + (\widetilde{\beta}_i(s') \Big\}$$

is the extrinsic $L(u_i)$ value、and denotes part of the $L(u_i)$ value that depends on the parity bits.

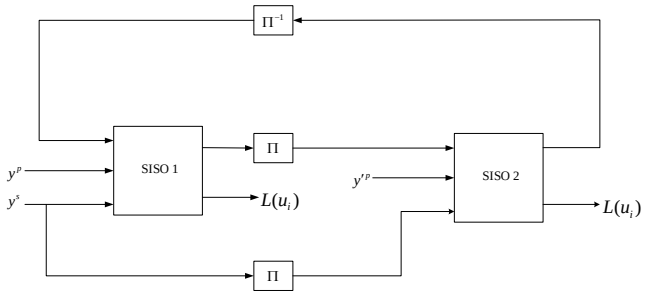

Figure 1: Turbo Decoder

The turbo decoder is shown in figure 1. To explain the iterative decoding process, the following assumptions are made

1 The component encoders of the Turbo encoder are RSC encoder with rate 1/2

2 The information bit sequence $\boldsymbol{u}_i = (u_1, u_2, ..., u_N)$ is fed into the first RSC encoder and outputs the first parity bit sequence$c^p = (c_1^p, c_2^p, ..., c_N^p)$.

3 The information bit sequence is fed into the interleaver and outputs、$\boldsymbol{u'}_i = (u'_1, u'_2, ..., u'_N)$ which is then fed into the second component encoder to produce the second parity bit sequence、$\boldsymbol{c'}^p = (c'^p_1, c'^p_2, ..., c'^p_N)$.

4 $\boldsymbol{u}_i, \boldsymbol{c}^p, \boldsymbol{c'}^p$ are BPSK modulated and transmitted over an AWGN channel.

5 The received sequence is、$\boldsymbol{y}_i^s, \boldsymbol{y}^p, \boldsymbol{y'}_i$ and becomes the input to the decoder.

The turbo decoding process is as follows

1 The input to the first component decoder is $(\boldsymbol{y}^s, \boldsymbol{y}^p)$ and relates to the first component encoder of the Turbo encoder.

2 using equation 17、$L(u_i)$ is calculated. For the first iteration、u_i is assumed to have equal probability and$L^{(a)}(u_i)$ is set to 0。

3 At the output of the first componet decoder, $L_c y_i^s$ is subtracted from $L(u_i)$ to yield $L_{12}^{(e)}(u_i)$. $L_{12}^{(e)}(u_i)$ is then interleaved and fed into the second component decoder as the value for $L^{(a)}(u_i)$.

4 The other inputs to the second component decoder are $(\boldsymbol{y'}^s, \boldsymbol{y'}^p)$ and are used to calculate $L_{21}^{(e)}(u_i)$.

5 $L_{21}^{(e)}(u_i)$ is deinterleaved、and fedback into the first component encoder as the new $L^{(a)}(u_i)$ value.

The iteration is either repeated for a predetermined number of times, or untill a certain condition is met. At the final iteration $L(u_i)$ (from the second component decoder)is deinterleaved and used to estimate the value ofu_i.

5.1 System Model

The system model used for the simulation is shown in figure 2. The conditons used for the various sytem blocks are given below。

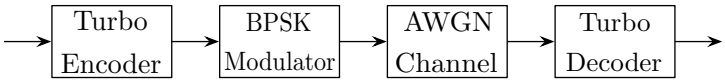

Figure 2: System Model

1. Turbo Encoder

a The input to the Turbo encoder is binary.

b The component encoders for the Turbo code are K=3,k=1, n=2 RSC encoders

c Quadratic, S-random and Quadratic permutation polynomial interleavers[5] are used. Each one has length ,2^m where the value for m is within the range $3-10$, with increments on 1.

d output rate of the Turbo encoder is 1/3.

2. BPSK Modulator

When the input to the modulator is bit 1 the output is +1, when the input is bit 0, the output is -1.

3. AWGN Channel

Zero mean white Gaussian Noise is used. The output SNR is within the range 1 -2 dB with step size of 0.5.

4. Turbo Decoder

Quadratic, S-random and Quadratic permutation polynomial interleavers are used. The Max-Log-MAP decoding algorithm is used for the iterative decoding

6 References

[**1**] John G. Proakis, Masoud Salehi. "Digital Communications", Fifth Edition,Chapter 8, McGraw-Hill
.

[**2**] Oscar Y. Takeshita, Member, IEEE, and Daniel J. Costello ,"New Deterministic Interleaver Designs for Turbo Codes",IEEE Trans. Inform. Theory, vol. 46,pp. 1988-2006,Nov. 2000

[**3**] L. C. Perez, J. Seghers, D. J. Costello, Jr., "A distance spectrum interpretation of turbo codes", IEEE Trans. Inform. Theory, vol. 42, pp. 1698-1709, Nov. 1996.

[**4**] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes", Proc. Intern. Conf. Communications (ICC), Geneva, Switzerland, pp. 1064- 1070, May 1993.

[**5**] Jing Sun, Oscar Y. Takeshita "Interleavers for Turbo Codes Using Permutation Polynomials over Integer Rings", IEEE Trans. Inform. Theory, vol. 51, pp. 101 - 119 Jan. 2005