

# Noncoherent Turbo Decoding

Jacob Vainappel, Einat Hardy, and Dan Raphaeli

**Abstract**—In this paper, we propose a new Turbo code noncoherent detection scheme based on a novel a-priori probability processor (APP). Using this method, many turbo encoders can be used without a change in the code or in the modulation, and without a significant change in the structure of the iterative decoder. Simulation results of this method are presented for various phase noise levels. We show a degradation of 0.7 dB from the coherent detection of the same codes.

## I. INTRODUCTION

Various methods have been proposed for noncoherent decoding of phase encoded modulations transmitted through additive white Gaussian noise (AWGN) channels which also add phase noise. Noncoherent decoding schemes are attractive in scenarios for which phase estimation and correction loop (such as a PLL), followed by a coherent decoder, may not achieve the adequate BER performance or the initial phase synchronization time.

Some methods have been proposed for noncoherent Turbo detection. Noncoherent turbo decoding of differential encoded symbols on a general phase noise is introduced by [1]. In this approach, the phase was assumed to be independent after  $L$  symbols, and the APP was derived accordingly. Another approach was taken in [2] by modeling the phase noise as a discrete Markovian process. In this approach, the new trellis allows to approximately compute the 1information bit probabilities, taking into account the phase transition probabilities. The disadvantage is the need to model or estimate the phase process which may vary in time and in between units.

In this paper we present a noncoherent sequence estimation decoder for Trellis and Turbo codes. This decoder is built under the assumption that the phase is constant during  $L/r$  symbols, where  $L$  and  $r$  denote the code rate and the number of source bits respectively. This decoder can be used without modifications in the encoder as long as it uses noncoherent non-catastrophic error correcting codes (ECC). After introducing noncoherent sequence estimation, we develop a “noncoherent” version of the Bahl-Cocke-Jelinek-Raviv (BCJR) [3], which is the basic building block of the Turbo decoder. To further improve the performance, we then create “phase linkage” between the two BCJR elements in the Turbo decoder.

A somewhat similar path is taken by [4] to develop a maximum a-posteriori probability (MAP) decoder for the noncoherent case. However, it does not cope with the phase relation between the two decoders.

## II. NONCOHERENT SEQUENCE ESTIMATION

Sequence decoding assumes a slow phase change. Thus a constant phase (or any known phase statistics) can be assumed in an observation of a sequence which is not too long. For linear modulations, assuming AWGN, the channel can be modeled by  $y_k = x_k e^{j\phi} + n_k$ , where  $\phi$  is a slowly changing random variable. That is, during one symbol transmission, the phase is considered to be constant. Provided that the ECC is a noncoherent-noncatastrophic code, the decoders can take advantage of this fact in order to resolve phase ambiguities.

Noncoherent codes have the property that, given any two information sequences long enough, their corresponding encoded symbol-sequences differ by more than just a phase shift. Assuming a constant phase associated with a sequence of received symbols  $Y$ , the transmitted source symbols  $X$  can be estimated by finding the maximum likelihood source symbols.

For (coherent) trellis decoding, the estimated source bits are the ones that form the highest probability path. Its probability is the product of all the branch probabilities along this path. The probability of each trellis branch is independent of any other branch. The latter, however, is not true for noncoherent decoding. Calculating the AWGN channel probability for each symbol independently is incorrect and will result in a loss of phase information (for example, the channel may shift the phase by  $180^\circ$  yielding incorrect probability of the PSK transmitted symbol).

However, employing the fact that a sequence of up to  $L$  information bits can be assumed to have equal phase, the channel probability can be computed by observing symbol-sequences instead of a symbol at a time.

It is clear that the slower the phase change, the larger the observed sequences can be, and thus, the performance can be improved. In addition, in order to fully exploit the phase information, any sequence should be observed in the calculations, thus sequence overlapping should be used (see figure 1).

## III. THE LIMITED-MEMORY BCJR ALGORITHM

The original BCJR algorithm [3] has been developed for the case of independent channel transition probabilities between any two symbols. The addition of a phase shift to the channel, relates any two symbols that are not too far apart from each other. Taking this into account, the BCJR components should be rendered accordingly.

A binary vector  $U = (u_1, \dots, u_N)$ ,  $u_k \in \{-1, +1\}$ , is trellis encoded to form a vector  $X$  of size  $N$  (the transition

$s_{k-1}$  to  $s_k$  outputs  $x_k$ ). The vector passes through a channel and results in a random vector  $Y$  of size  $N$ , where the channel probability  $\Pr(Y|X)$  is known. Note, that given  $X$ , the received symbols  $Y$  are unnecessarily independent. We wish to find  $\Lambda(u_k)$ , the log-likelihood ratio (LLR) estimation of  $u_k$

$$\begin{aligned}\Lambda(u_k) &\triangleq \log \left( \frac{\Pr(u_k = +1|Y)}{\Pr(u_k = -1|Y)} \right) \\ &= \log \left( \frac{\Pr(Y, u_k = +1)/\Pr(Y)}{\Pr(Y, u_k = -1)/\Pr(Y)} \right) \\ &= \log \left( \frac{\sum_{u^+} \Pr(s_{k-1} = s', s_k = s, Y)/\Pr(Y)}{\sum_{u^-} \Pr(s_{k-1} = s', s_k = s, Y)/\Pr(Y)} \right)\end{aligned}\quad (1)$$

Where  $u^+$  and  $u^-$  are the set  $(s', s)$  of trellis transitions  $s' \rightarrow s$  produced by the input  $u = +1$  and  $u = -1$  respectively.

$$\begin{aligned}\Pr(s_{k-1} = s', s_k = s, Y) &= \Pr(s, s', Y_1^{k-1}, y_k, Y_{k+1}^N) \\ &= \underbrace{\Pr(Y_{k+1}^N | s, s', Y_1^k)}_{\beta_k(s, s')} \cdot \underbrace{\Pr(s, y_k | s', Y_1^{k-1})}_{\gamma_k(s, s')} \cdot \underbrace{\Pr(s', Y_1^{k-1})}_{\alpha_{k-1}(s')}\end{aligned}\quad (2)$$

$$\begin{aligned}\gamma_k(s, s') &\triangleq \Pr(s, y_k | s', Y_1^{k-1}) \\ &= \Pr(y_k | s, s', Y_1^{k-1}) \cdot \Pr(s | s', Y_1^{k-1}) \\ &= \Pr(y_k | x_k, Y_1^{k-1}) \cdot \underbrace{\Pr(s | s')}_{\Pr(u_k)}\end{aligned}\quad (3)$$

The  $\alpha_k(s)$  term is computed recursively

$$\begin{aligned}\alpha_k(s) &\triangleq \Pr(s, Y_1^k) = \sum_{s'} \Pr(s, s', Y_1^k) \\ &= \sum_{s'} \Pr(s, y_k | s', Y_1^{k-1}) \cdot \Pr(s', Y_1^{k-1}) \\ &= \sum_{s'} \gamma_k(s, s') \alpha_{k-1}(s')\end{aligned}\quad (4)$$

With the initial conditions

$$\alpha_1(s = 0) = 1 \quad \text{and} \quad \alpha_1(s \neq 0) = 0 \quad (5)$$

The  $\beta_k(s, s')$  term cannot be computed recursively unless we have a channel for which the knowledge of  $s'$  does not affect (or has little effect on)  $\beta$ . For such a channel,  $\beta_k(s, s') \approx \beta_k(s)$ , thus allowing recursive formulation

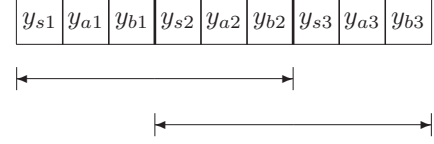


Fig. 1. Overlapping of symbol blocks.  $L = 2$ ,  $r = 1/3$ .  $y_s$  are the systematic symbols,  $y_a, y_b$  are the parity symbols of the Turbo encoder.

(note that  $s$  matches  $s_k$  and  $s'$  matches  $s_{k-1}$ )

$$\begin{aligned}\beta_{k-1}(s') &\triangleq \Pr(Y_k^N | s', Y_1^{k-1}) \\ &= \sum_s \Pr(Y_k^N, s | s', Y_1^{k-1}) \\ &= \sum_s \Pr(Y_{k+1}^N | s, s', Y_1^k) \cdot \Pr(y_k, s | s', Y_1^{k-1}) \\ &= \sum_s \beta_k(s) \gamma_k(s, s')\end{aligned}\quad (6)$$

With the initial conditions

$$\beta_N(s = 0) = 1 \quad \text{and} \quad \beta_N(s \neq 0) = 0 \quad (7)$$

Comparison of the modified BCJR to the original BCJR reveals that the only computational change is that the original  $\Pr(y_k | x_k)$  transforms into  $\Pr(y_k | x_k, Y_1^{k-1})$  in the  $\gamma$  term. All the other changes (the different definition of  $\alpha, \beta$ ) don't affect the recursive formula.

This probability function is the only needed knowledge about the channel. It is given the notation

$$\mathcal{P}_k(Y|X) \triangleq \Pr(y_k | x_k, Y_1^{k-1}) \quad (8)$$

Finally, substituting (2) in (1) yields

$$\Lambda(u_k) = \log \left( \frac{\sum_{u^+} \alpha_{k-1}(s') \gamma_k(s, s') \beta_k(s)}{\sum_{u^-} \alpha_{k-1}(s') \gamma_k(s, s') \beta_k(s)} \right) \quad (9)$$

#### IV. NONCOHERENT BCJR IMPLEMENTATION

The limited memory BCJR algorithm can be satisfied by defining a new overlapped observations decoding scheme

$$\begin{aligned}\mathbb{S}_k &= \{s_k, s_{k-1}, s_{k-2}, \dots, s_m\} \\ \mathbb{Y}_k &= \{y_k, y_{k-1}, y_{k-2}, \dots, y_m\} = Y_m^k \\ \mathbb{X}_k &= \{x_k, x_{k-1}, x_{k-2}, \dots, x_m\} = X_m^k\end{aligned}\quad (10)$$

Where  $m = k - L + 1$ . For such a scheme,

$$\begin{aligned}\beta_{k-1}(\mathbb{S}, \mathbb{S}') &= \Pr(\mathbb{Y}_k^N | \mathbb{S}, \mathbb{S}', \mathbb{Y}_1^{k-1}) \\ &= \Pr(\mathbb{Y}_k^N | \mathbb{S}, s_{m-1}, \mathbb{Y}_1^{k-1})\end{aligned}\quad (11)$$

The  $s_{m-1}$  term can be neglected under the approximation of (12). By observing multiple trellis states at once,

the original code trellis can be augmented to a new trellis, in which each state encapsulates the last original  $L$  states. The number of states in the new trellis is larger by a factor of  $2^{L-1}$  than that in the original one. Hence, the BCJR decoder observes  $\mathbb{Y}_k$  for each augmented trellis branch  $\mathbb{S}_{k-1} \rightarrow \mathbb{S}_k$  (observing  $L/r$  symbols instead of  $1/r$ ).

We now turn to the development of (8). Observation length of  $L$  bits instead of one bit expands it to

$$\begin{aligned} \mathcal{P}_k(\mathbb{Y}|\mathbb{X}) &\triangleq \Pr(Y_m^k | X_m^k, Y_1^{k-1}) \\ &= \Pr(y_k | X_m^k, Y_1^{k-1}) \\ &\approx \Pr(y_k | X_m^k, Y_m^{k-1}) \end{aligned} \quad (12)$$

This approximation assumes that symbols that are at least  $L/r$  apart are independent (else, a larger  $L$  could be taken, and better results would be achieved).

Note that  $y_k$  is affected by the two parameters  $x_k$  and  $\phi$ .

$$\begin{aligned} \Pr(y_k | X_m^k, Y_m^{k-1}) &= \\ &\int_0^{2\pi} \Pr(y_k | x_k, \phi) \Pr(\phi | X_m^{k-1}, Y_m^{k-1}) d\phi \end{aligned} \quad (13)$$

This justifies the approximation of (12). The dependency of  $y_k$  at the omitted  $Y_1^{m-1}$  contributes only to the phase estimation term of (13). Hence,  $L$  is chosen as the estimation length.

$$\begin{aligned} \Pr(\phi | X_m^{k-1}, Y_m^{k-1}) &= \\ \Pr(Y_m^{k-1}, X_m^{k-1} | \phi) \frac{\Pr(\phi)}{\Pr(Y_m^{k-1}, X_m^{k-1})} &= \\ \Pr(Y_m^{k-1} | X_m^{k-1}, \phi) \Pr(X_m^{k-1}, \phi) \frac{1/2\pi}{\Pr(Y_m^{k-1} | X_m^{k-1}) \Pr(X_m^{k-1})} &= \\ = \frac{\Pr(Y_m^{k-1} | X_m^{k-1}, \phi)}{2\pi \Pr(Y_m^{k-1} | X_m^{k-1})} \end{aligned} \quad (14)$$

Substituting this into (13)

$$\begin{aligned} \Pr(y_k | Y_m^{k-1}, X_m^k) &= \frac{1}{2\pi \Pr(Y_m^{k-1} | X_m^{k-1})} \\ &\cdot \int_0^{2\pi} \Pr(y_k | x_k, \phi) \Pr(Y_m^{k-1} | X_m^{k-1}, \phi) d\phi \end{aligned} \quad (15)$$

Noticing that the two probabilities inside the integral are Gaussian, independent, and share the same  $\phi$ . Therefore, the integrand becomes  $\Pr(Y_m^k | X_m^k)$ , and thus

$$\Pr(y_k | X_m^k, Y_m^{k-1}) \propto \frac{\Pr(Y_m^k | X_m^k)}{\Pr(Y_m^{k-1} | X_m^{k-1})} \quad (16)$$

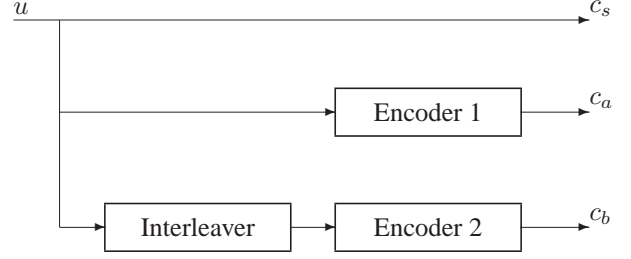


Fig. 2. Turbo Encoder: output codeword  $c$  consists of three (possibly two) outputs. One systematic, two out of the constituent encoders.

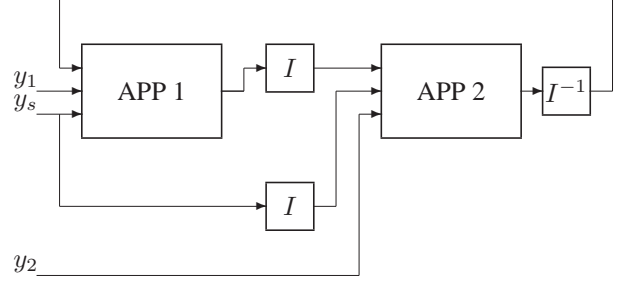


Fig. 3. Turbo Decoder. The  $I$  and  $I^{-1}$  blocks stand for interleaver and de-interleaver respectively.

The noncoherent probability  $\Pr(Y|X)$  for an AWGN channel with unknown constant phase is

$$\begin{aligned} P_{\text{NC}}(Y|X) &\triangleq \Pr(Y|X) = \int_0^{2\pi} \Pr(Y|X, \phi) \Pr(\phi) d\phi \\ &= \frac{1}{(2\pi\sigma^2)^N} \exp\left(\frac{-1}{2\sigma^2} \sum_{i=1}^N [|x_i|^2 + |y_i|^2]\right) \\ &\quad \cdot I_0\left(\frac{1}{\sigma^2} \left| \sum_{i=1}^N x_i y_i^* \right|\right) \end{aligned} \quad (17)$$

Where  $I_0(\cdot)$  is the modified Bessel function. Note that the  $\exp(\cdot)$  term is a not a function of  $X$  for equal energy modulations, such as PSK. In this case, it simplifies to

$$P_{\text{NC}}(Y|X) \propto I_0\left(\frac{1}{\sigma^2} \left| \sum_{i=1}^N x_i y_i^* \right|\right) \quad (18)$$

## V. TURBO DECODING

The classic Turbo coding is introduced by [5]. It consists of the encoder of figure 2 and the iterative decoder of figure 3. Each one of the two APPs (alternatively, they are called MAPs) in the decoder operates on its own symbol set. That is, one operates on the received signals  $Y_s$  and  $Y_a$ , whereas the other estimator operates on  $Y_s$  and  $Y_b$ . These estimators are implemented using the BCJR algorithm.

A straightforward implementation of a noncoherent Turbo decoder is replacing the two BCJR blocks with noncoherent ones as described in the previous sections. Note that for the noncoherent case, the interleaved systematic input of the second encoder must be omitted, otherwise the equal phase assumption in each observation does not hold. In addition, the encoder should output  $(L-1)/r$  symbols preamble to allow correct trellis branch probability computation of the first information bits (each augmented trellis branch depends on the  $L - 1$  previous source bits).

Such a scheme yields a good performance but it can still be improved by observing that no phase information is shared among the two BCJRs.

This can be shown as follows: Assume a constant phase over the entire sequence, and shift the phase of  $Y_b$  (all of the symbols entering the second decoder) by a constant amount. Due to the noncoherent channel probability, the second output remains unchanged. Furthermore, the first decoder is not aware of this phase change. From this example, it is clear that in the current split input implementation ( $Y_s, Y_a$  are entered to the first decoder, and  $Y_b$  is entered to the second one), essential phase information is lost.

However, sharing phase information can improve the results by using  $Y_s$  non-interleaved as an input to the second decoder. Its value on the trellis is unknown, but its current probability estimation (its extrinsic probability) is the current input LLR. This way, (12) should be computed according to  $\Pr(y_k, y_{s,k} | X_m^k, Y_m^k, u_k)$  where  $y_{s,k}$  is the systematic interleaved  $k$  symbol, thus rendering (16) to

$$\Pr(y_k, y_{s,k} | X_m^k, Y_m^k, u_k) = \sum_{i \in \{0,1\}} \Pr(u_k = i) \frac{P_{\text{NC}}(y_{s,k}, Y_m^k | u_k = i, X_m^k)}{P_{\text{NC}}(Y_m^k | X_m^k)} \quad (19)$$

This probability can be augmented to consider more symbols from the other decoder.

## VI. SIMULATION RESULTS

The Turbo encoder simulated is an  $r = 1/2$  code comprised of two  $(7, 5)$  recursive encoders, a random interleaver, a source block of  $N = 4096$  bits, and a BPSK modulator (a similar structure as shown in [5]). A comparison of the coherent decoder versus the noncoherent decoder types is shown in figure 4. The improved noncoherent decoder uses one non-interleaved systematic input added to the second APP decoder (as explained in section V).

The performance in a phase noise environment is shown for random walk and for frequency deviation. In random walk phase noise, the phase at each symbol is  $\phi_i = \phi_{i-1} + \theta_i$ , where  $\theta \sim N(0, \sigma)$ . Figure 5 shows the performance of the improved decoder for  $L = 3$  and various phase noise levels ( $\sigma$ ). The simulation shows that  $\sigma = 12^\circ$  causes a degradation of 0.4 dB.

Results of frequency deviation,  $\theta_i = \Delta = \text{constant}$ , are shown in figure 6.

## VII. CONCLUSIONS

We have presented a soft-output noncoherent decoding scheme for convolutional and Turbo codes. By generalizing the BCJR algorithm and augmenting the decoder's trellis, we were able to observe a larger symbol sequence at a time. Such an augmentation satisfies the assumption in made in (12). For turbo codes, further improvement is made by creating a phase "linkage" between the APP decoders, yet without adding a soft phase output by each decoder.

For observation blocks of size  $L = 4$ , the simulation shows a performance degradation of about 1.5 dB in respect to a coherent decoder and perfectly known phase. Coupling the phase between the decoders decreases the gap to about 0.7 dB.

## REFERENCES

- [1] Hoeher, and Lodge "Turbo DPSK: Iterative Differential PSK Demodulation and Channel Decoding". *IEEE Transactions on Communications*, VOL. 47, No. 6, June 1999
- [2] Peleg, and Shamai "On Coded and Interleaved Noncoherent Multiple Symbol Detected MPSK" *European Transaction on Telecommunications and Related Technologies (ETT)*, Vol. 10, No. 1, pp. 65-74, January/February 1999
- [3] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", *IEEE Trans. Inf. Theory*, p. 284-287, Mar. 1974.
- [4] Colavolpe, Raheli, and Ferrari "A Noncoherent Soft-Output Decoding Algorithm for Coded Linear Modulations". *IEEE ICC '99*
- [5] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes", *Proc. 1993 Int. Conf. Comm.*, p. 1064-1070
- [6] D. Raphaeli "Noncoherent Coded Modulation". *IEEE Transactions on Communications*, vol. 44, February 1996, pp. 172-183.

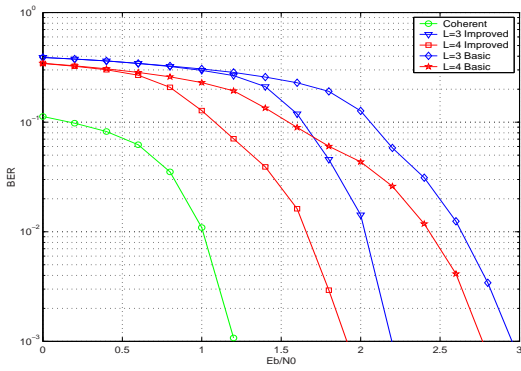


Fig. 4. Performance comparison of the coherent decoder versus the basic noncoherent decoder and the phase-coupled improved decoder (where the systematic input is fed into the second decoder) for two observation lengths:  $L = 3, 4$ . For the noncoherent decoders, the phase added is constant.

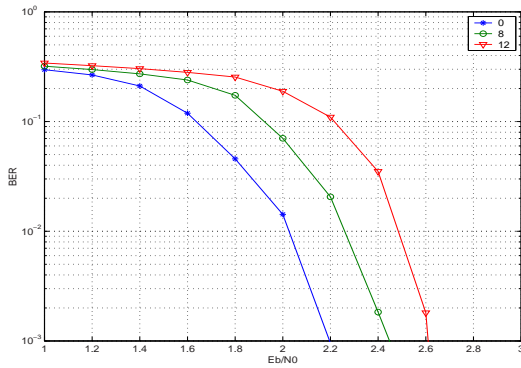


Fig. 5. Performance of the improved decoder with  $L = 3$  in random walk Gaussian noise. The legend shows the various  $\sigma$  values.

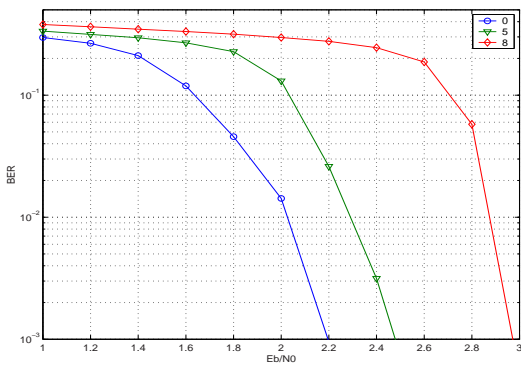


Fig. 6. Performance of the improved decoder with  $L = 3$  in a frequency shift. The legend shows the various  $\Delta$  values in degrees.