An Iterative, Turbo-like Approach to Convolutional Decoding

Laurent SCHMALEN, Marc ADRAT

Institute of Communication Systems and Data Processing, RWTH Aachen University,
Templergraben 55, 52056 Aachen, Germany

laurent@ind.rwth-aachen.de, marc@ind.rwth-aachen.de

Abstract. *In this paper we present a new method to decode convolutional codes. The method is based on the Turbo principle which was originally proposed to decode parallel concatenated convolutional codes. It becomes possible because of a reinterpretation of the encoder structure. The newly interpreted code is decoded using an iterative scheme with information exchange between sub-decoders.*

In a second part of the paper, the new method is applied to the decoding of the GSM Full Rate Channel Code. The simulation results show that each iteration can improve the decoding performance while the original BCJR algorithm marks the upper limit.

Keywords

Channel Coding, Convolutional Decoding, GSM Full Rate Codec, Maximum-A-Posteriori BCJR Algorithm, Turbo-Decoding, *L*-Values.

1. Introduction

Whenever in digital mobile communication data transmission is affected by channel noise, channel coding is indispensable. In practice, frequently, convolutional codes are applied to protect the source data and correct some possibly given transmission errors. The decoding is usually performed using the Viterbi algorithm [1] or the BCJR algorithm [2]. The Viterbi decoder determines the optimal bit sequence in the maximum likelihood sense and the BCJR algorithm provides the optimal sequence of bits in the *maximum-a-posteriori* sense. Recently, the use of parallel concatenated interleaved convolutional codes, named *Turbo-Codes* [3], and their iterative decoding has allowed further improvements in error correcting coding. The question arises if the iterative decoding process used in *turbo-decoding* will be applicable to the decoding of “normal” convolutional codes. The goal of this paper will be to find an answer to that question.

In Section 2 the encoder structure of a conventional convolutional code is reinterpreted as to allow iterative decoding subsequently followed by a description of the decoder. The single elements composing the decoder will also be presented in this section. The newly presented decoding scheme will finally be applied to the GSM full rate channel code [5], [6].

In Section 3 the simulation results are presented. The simulations are an application of GSM channel decoding.

2. Iterative Decoding of Convolutional Codes

The iterative decoding system presented in this paper is explained using the convolutional encoder specified by the GSM full rate codec [6]. This code is a rate ½ code and is characterized by the octal generator polynomials **G**(23,33). This encoder is depicted in Fig. 1. As can be seen, the encoder possesses 4 memory elements and has therefore got a constraint length of 5. This implies a trellis representation with $2^4 = 16$ states.

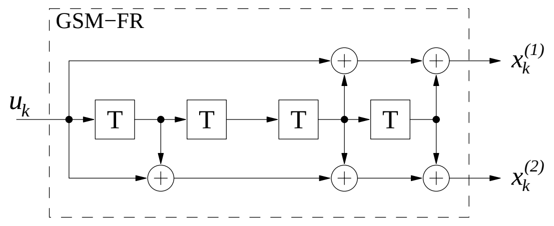

Fig. 1. Convolutional encoder used in the GSM full rate codec

The encoding process can be represented differently if the encoder is drawn as in Fig. 2. The input is fed to two identical encoders. The outputs of these encoders are punctured so that the upper output of encoder 1 and the lower output of encoder 2 are used. If an optional interleaver Π is inserted in front of the input of encoder 2, this representation is more or less equivalent to the *turbo-coding* scheme presented in [3] (except for the used generator polynomials). In this work however, the interleaver is omitted so as to decode “normal” convolutional codes.

Thus, it was shown, that a convolutional code can be represented by the parallel concatenation of two appropriate punctured convolutional encoders. In the following subsections, a decoding method using this representation is presented.

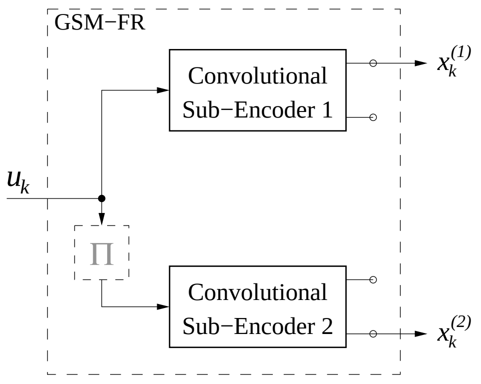

Fig. 2. Reinterpretation of the GSM-FR channel encoder

2.1 The SISO Module

The decoding makes use of a so-called SISO (Soft Input Soft Output) module [4] which is presented in this section. The SISO Module presented in Fig. 3 has two inputs, one accepting the received channel values $L(\boldsymbol{y}_k)$ and one accepting *a-priori* knowledge about the data bits $L(u_k)$.

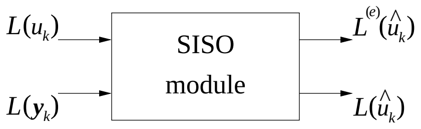

Fig. 3. Soft Input Soft Output module used for iterative decoding

In this paper, the transmission of the code symbols is performed over an AWGN (Additive White Gaussian Noise) channel with BPSK (Binary Phase Shift Keying) mapping. In that case, the received symbols are:

$$y_k^{(i)} = a \cdot \tilde{x}_k^{(i)} + n \tag{1}$$

with the bipolar mapped code symbol:

$$\tilde{x}_k^{(i)} = \begin{cases} -1 & \text{if } x_k^{(i)} = 1 \\ +1 & \text{if } x_k^{(i)} = 0 \end{cases} \tag{2}$$

In equation (1), the factor a denotes the attenuation of the channel and n the white Gaussian noise process. In this case, the L-values of the received code symbol equal [4]:

$$L\left(y_k^{(i)}\right) = L_c \cdot y_k^{(i)} \tag{3}$$

with

$$L_c = 4a\frac{E_s}{N_0} \tag{4}$$

In equation (4), E_s represents the transmission energy of a symbol bit and $N_0/2$ is the power density of the channel noise n. The actual decoding is performed using the BCJR algorithm [1] that has been adapted to log-likelihood calculations [4] because of numerical and efficiency reasons. The *a-posteriori* L-values of the estimated data bits $\hat{u}_k$ are given by the following equations:

$$L(\hat{u}_k) = L(u_k) + L^{(e)}(\hat{u}_k) \tag{5}$$

$$L^{(e)}(\hat{u}_k) = \log \frac{\sum\limits_{\substack{(s_{k-1},s_k)\\u_k=+1}} \alpha_{k-1}(s_{k-1})\gamma_k^{(e)}(s_{k-1},s_k)\beta_k(s_k)}{\sum\limits_{\substack{(s_{k-1},s_k)\\u_k=-1}} \alpha_{k-1}(s_{k-1})\gamma_k^{(e)}(s_{k-1},s_k)\beta_k(s_k)} \tag{6}$$

The terms s_k in equation (6) represent the states in the trellis-representation and (s_{k-1}, s_k) denotes a state transition. The coefficients $\alpha_{k-1}(s_{k-1})$ are obtained by the forward recursion in equation (7) and the $\beta_k(s_k)$ are obtained using the backward recursion in equation (8). The forward recursion is initialized with $\alpha_0(s_0 = 0) = 1$ and $a_0(s_0 \neq 0) = 0$ whereas the backward recursion is initialized with $\beta_N(s_N = 0) = 1$ and $\beta_N(s_N \neq 0) = 0$. N represents the last state of a transmitted block.

$$\alpha_k(s_k) = \sum_{s_{k-1}} \gamma_k(s_{k-1},s_k)\alpha_{k-1}(s_{k-1}) \tag{7}$$

$$\beta_{k-1}(s_{k-1}) = \sum_{s_k} \gamma_k(s_{k-1},s_k)\beta_k(s_k) \tag{8}$$

Whenever a branch transition from state s_{k-1} to state s_k exists, its transition probability is given by

$$\gamma_k(s_{k-1},s_k) = e^{\frac{1}{2}u_k L(u_k)}\gamma_k^{(e)}(s_{k-1},s_k) \tag{9}$$

with the extrinsic transition probability

$$\gamma_k^{(e)}(s_{k-1},s_k) = \exp\left(\frac{1}{2}\sum_{i=1}^{2} L_c y_k^{(i)} x_k^{(i)}\right) \tag{10}$$

Note that equations (9) and (10) are only valid in case of a rate ½ feed-forward convolutional code.

2.2 Iterative Decoding

The actual iterative decoding is performed using the system presented in Fig. 4.

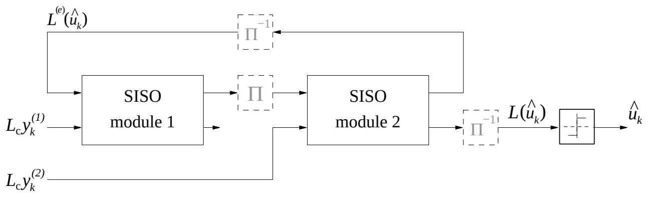

Fig. 4. Iterative convolutional decoder

The received values that have been encoded with the upper encoder in Fig. 2, i.e. the first bits of the output symbols, are fed into the first SISO module. The extrinsic output of that module is used as *a-priori* input of the second SISO module which accepts as channel input the received values of the lower encoder. The extrinsic output of the second SISO module is then fed back to the *a-priori* input of the first module. Note that the extrinsic transition probability in equation (10) can be simplified to:

$$\gamma_k^{(e)}(s_{k-1},s_k) = \exp\left(\frac{1}{2}L_c y_k^{(i)} x_k^{(i)}\right) \tag{11}$$

with *i* equals 1 in case of module 1 and *i* equals 2 in case of the module 2. Thus only one bit of the code symbol is used for branch transition calculations.

After a given number of iterations, the output of the estimated bits of the second decoder is used to perform the decision on the received bits:

$$\hat{u}_k = \begin{cases} 1 & \text{if } L(\hat{u}_k) \le 0 \\ 0 & \text{if } L(\hat{u}_k) > 0 \end{cases} \qquad (12)$$

The decoding scheme is identical to the one used to decode parallel concatenated convolutional codes (turbo-codes) but without any interleavers. Though, the interleavers are drawn in a dashed way in Fig. 4. This is to indicate that they are not used in this case, but that an already present turbo-decoding module could also be used to decode convolutional codes using the presented method.

3. Simulation Results

The simulation was carried out using the rate 1/2 convolutional encoder specified by the GSM full rate codec [6]. The generator polynomials of the code are **G**(23,33) (see Fig. 1) in octal notation. As in GSM, the block length was chosen to 185 data bits plus 4 zero bits for termination purposes. Note that in GSM only the 185 most important (class 1) bits are channel encoded. The remaining 78 less significant bits (class 2) are not protected.

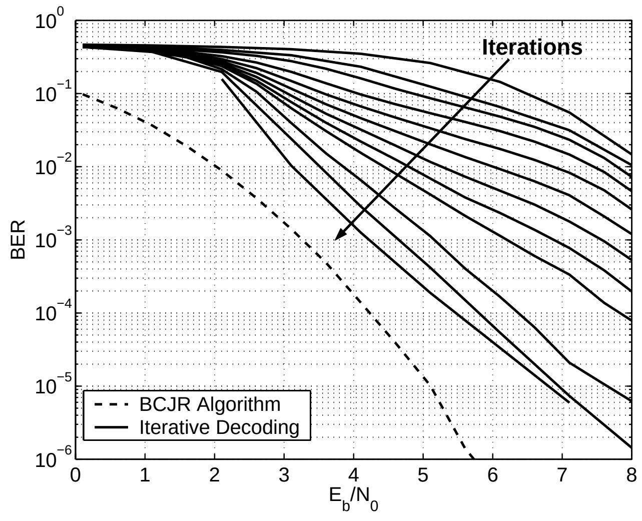

Fig. 5. Simulation results for $I = 1, 2, 3, 5, 7, 9, 11, 13, 15, 21, 30$ and 100 decoding iterations

As can be seen in Fig. 5, the iterative decoding presented in Section 2 yields some interesting results. Each decoding step improves the decoding performance in terms of BER. Thus an increasing number of iterations will slowly lead to a better decoding result. After $I = 100$ decoding iterations, the result of the BCJR algorithm is approached by approximately 1dB. A higher number of iterations may even further increase the decoding performance as no convergence can yet be seen. Unfortunately, the performance of the *maximum-a-posteriori* (BCJR) algorithm is not reached. The complexity is increased by factor $2I$, with I representing the number of iterations. The factor 2 is due to the two SISO operations in each iteration.

4. Conclusions

We have shown that convolutional codes can be decoded using a turbo-decoder with an omitted interleaver. The decoding performance approaches the result of the BCJR algorithm with an increasing number of decoding iterations. This is an interesting result because the omitted interleaver is usually a key element in iterative decoding.

The increased complexity may be reduced by the use of less complex SISO algorithms, e.g. the M-BCJR [7] or the SOMA [8] algorithms.

Acknowledgements

The authors would like to thank the head of the Institute of Communication Systems and Data Processing, Prof. P. Vary. The underlying work of this paper is the result of a student research project which was carried out at his institute.

References

[1] FORNEY, G. D. Jr., The viterbi algorithm, *Proceedings of IEEE*, vol. 61, p. 268 – 278, Mar. 1973.

[2] BAHL, L. R., COCKE, J., JELINEK, F., RAVIV, J. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inform. Theory*, p. 284 – 287, Mar. 1974.

[3] BERROU, C., GLAVIEUX, A., Near optimum error correcting coding and decoding: turbo-codes. *IEEE Trans. Commun.*, vol. 44, no. 10, p. 1261 – 1271, Oct. 1996.

[4] HAGENAUER, J., OFFER, E., PAPKE, L. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inform. Theory*, vol. 42, p. 429 – 445, Mar. 1996.

[5] ETSI TC-SMG, Recommendation 6.10: GSM Full Rate Speech Transcoding, 1992.

[6] ETSI TC-SMG, Recommendation 5.03: Digital Cellular Telecommunications System (Phase 2+); Channel Coding, 1999.

[7] FRANZ, V., ANDERSON, J. B., Concatenated decoding with a reduced-search BCJR algorithm. *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, p. 186 – 195, Feb. 1998.

[8] WONG, K. K. Y., MCLANE, P. J., Bi-directional soft-output M-algorithm for iterative decoding. *IEEE Commun. Soc. Mag.*, 2004.