Improved Turbo Decoders for Signals with Time and Phase Offset on AWGN Channels

Bartosz Mielczarek, Arne Svensson
Communication Systems Group
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
PH: +46 31 772 1763 FAX: +46 31 772 1748,
e-mail:Bartosz.Mielczarek@s2.chalmers.se, Arne.Svensson@s2.chalmers.se

ABSTRACT

This paper presents behaviour of a turbo coding scheme when the synchronization parameters are not perfectly estimated. We propose improved decoding algorithms for the situations when the timing and phase error are given by the Gaussian probability distribution. It is shown that relatively simple operations can improve the bit error probability without the need of reconstructing the classical decoding algorithm and increasing transmitted power.

INTRODUCTION

The future wireless telecommunication systems will be required to transmit significantly more data than today which can lead to enormous increase of costs of connections if no new solutions are proposed.

The most important factors for efficiency of wireless transmission are bandwidth occupation and power required for successful transmission of data. The bandwidth occupation can be reduced by using advanced modulation techniques but also by reducing the amount of additional pilot signals needed for synchronization of the receivers. The power requirements can be reduced by using special coding techniques - the most prominent of them being the turbo coding, which is almost able to reach the theoretical Shannon channel capacity limit. The problem lies in the trade-off between those two requirements, since reducing the power of the signal leads to more difficult synchronization and increased number of pilot symbols.

In our paper we propose algorithms aiming at improving performance of turbo-coded systems, without restoring to additional pilot symbols.

TURBO CODES AND SYNCHRONIZATION

Good synchronization is essential in all digital wireless communication. The incoming HF signal needs to be downconverted to lower frequency and sampled properly in order to minimize intersymbol interference [6]. Such conversion requires exact knowledge of carrier frequency, carrier phase and symbol period.

Turbo codes (as shown in [8]) are not robust against timing and phase mismatch - even moderate offsets can cause performance losses which effectively counteract the gains of the turbo scheme, making it a viable option to use convolutional codes which have fewer problems with synchronization.

MODELLING OF NON-PERFECT SYNCHRONIZATION

The turbo coding schemes (rate 1/2 with QPSK modulation) usually assume that the signal after the AWGN channel has the form of

\[ y_k = c_k + n_k = y_k^p + j \cdot y_k^p \]  

(1)

where \( y_k \) is the complex received signal. The coded sequence is given by

\[ c_k = u_k + j \cdot x_k^p \]  

(2)

where \( u_k \) is the systematic bit and \( x_k^p \) the parity bit. The received bits corresponding to \( u_k \) and \( x_k^p \) are denoted \( y_k^s \) and \( y_k^p \), respectively. \( n_k \) is the white, additive, Gaussian complex noise with \( E[|n_k|^2] = 2\sigma_0^2 \).

In reality, however, the samples can be modelled as

\[ y_k = a(\varepsilon) e^{j\phi} c_k + n_{ISI} + n_k \]  

(3)

where \( a(\varepsilon) \) is a real scaling factor caused by non-ideal sampling, with the normalized offset of \( \varepsilon \), \( \phi \) is the phase offset of the carrier and \( n_{ISI} \) the additive distortion caused by intersymbol interference due to the timing offset. It can be shown empirically that \( n_{ISI} + n_k \) for low SNR values can be modelled as approximately Gaussian, thus giving

\[ y_k = a(\varepsilon) e^{j\phi} c_k + n' \]  

(4)

where \( E[|n'|_k^2] = 2(\sigma_0^2 + \sigma_{ISI}^2(\varepsilon)) = 2\sigma^2(\varepsilon) \).
The parameter $a(\epsilon)$ depends on the shape of the transmit and receive filters. Here we assume raised-cosine filtering which leads to (4)

$$a(\epsilon) = \frac{\text{sinc}(\epsilon) \cos(\alpha \pi \epsilon)}{1 - 4\alpha^2 \epsilon^2}$$

where $\alpha$ is the roll-off factor.

Figure 1 shows an example of such a distribution for a given timing offset and SNR value. Decreasing the SNR will give better correspondence to the Gaussian distribution, while increasing the timing offset will have the opposite effect - our simulations assume however, that large offsets are unlikely so the shapes not change.

The turbo coding schemes use the MAP approach for detection of data. The most commonly used decoding algorithm is the BCJR-MAP algorithm ([2]) which computes the soft bits using the log likelihood ratio (LLR) (see [2])

$$L(u_k|Y) = \log \frac{P(u_k=+1|Y)}{P(u_k=-1|Y)}$$

The LLR is calculated as (7)

$$L(u_k|Y) = \log \left( \frac{\sum_{S'} \tilde{a}_{k-1}(s') \cdot \gamma_k(s',s) \cdot \tilde{\beta}_k(s)}{\sum_{S'} \tilde{a}_{k-1}(s') \cdot \gamma_k(s',s) \cdot \tilde{\beta}_k(s)} \right)$$

where $L(u_k)$ is the value of the soft decoded bit $k$, $\gamma_k(s',s)$ and $\tilde{\beta}_k(s)$ are the recursively calculated probabilities of arriving at state $s'$ computed from the start and the end of the trellis, respectively, and finally $\gamma_k(s',s)$ is the apriori probability of the transition between states $s'$ and $s$. $\gamma_k(s',s)$ is given by

$$\gamma_k(s',s) = p(u_k)p(y_k|u_k) \propto \exp \left( \frac{1}{2} u_k(L'^c(u_k) + L_c y_k^T y_k^r) \right) \exp \left( \frac{1}{2} L_c y_k^r y_k^T \right)$$

where $L'^c(u_k)$ and $L_c = 4E_b/N_0$ are the extrinsic information about bit $k$ and the channel reliability factor of the decoder, respectively. The numerator of the MAP equation includes all the transitions caused by $u_k = +1$ ($S'^T$) and the denominator the transitions caused by $u_k = -1$ ($S'$), where $Y$ is the whole received sequence.

The MAP algorithm must be provided with exact ratio of signal to noise power, in order to give optimum performance (see [3]).
ALGORITHM MODIFICATION FOR TIMING ERRORS

With a timing offset, the actual channel reliability factor is given by

\[ L_c(\epsilon) = \frac{2E_c \sin^2(\epsilon) \cos^2(\alpha \epsilon)}{(\sigma_c^2 + 1.8049 \epsilon^2)(1 - 4\alpha^2 \epsilon^2)^2} \]  \hspace{1cm} (10)

In figure 3 we show \(10\log(L_c(\epsilon) - L_c(0))\), the error in channel reliability due to a timing offset. It can be seen that the actual channel reliability parameter for a large timing offset can be as low as 7dB under the ideal value. Such large differences are, however, rare and usually they should not be larger than 2-3dB.

Although channel reliability overestimation is less detrimental for the turbo decoder than underestimation (see [3]), even a small overestimation of the channel reliability factor reduces the performance as it can be seen from the figure 4. Here, the actual performance of the turbo coding scheme with non-perfect timing synchronization is presented together with the modelled behaviour of such code.

The modelled curves were obtained using equations (4), (6) and (10) to calculate the actual channel signal-to-noise ratio value for each offset and using the simulated ideal curve to get partial BER at each of the offsets. Finally the partial BER values were weighted according to the pdf of the timing error, thus forming a mean BER value for each given standard deviation of the timing error.

The third set of curves presents the behaviour of the turbo decoder assuming that at each timing offset outcome, the decoder knew exactly the channel reliability value. Those curves are almost identical with the theoretical curves, which suggests that the performance loss is due only to the channel reliability mismatch and confirms validity of our model.

As it can be seen the loss caused by overestimation of the channel reliability factor can be as high as 0.5 dB in a realistic case of moderate timing error. For small errors the difference is negligible, and for large ones, the overall loss in performance is too big to be considered.

The obvious approach to solve the problem would be to determine the actual channel reliability value for the given timing offset and for each block of data modify the \(L_c\) accordingly. As can be seen from figure 4, such an approach should be likely to reach the theoretical performance of the code. Unfortunately, such estimation can be rather imprecise since the systems operate at very low SNR.

There is, however, a possibility of large simplification of such a scheme, since the overestimation of the channel reliability value will not influence the behaviour of the code significantly when it is small. That gives the possibility of a simple thresholding operation - if the estimated timing offset is large, the channel reliability factor will be reduced from the nominal value. The timing offset can be simply estimated using squared values of the incoming

\[ \frac{1}{2}E(|y|^2) = a^2(\epsilon)E_c + \sigma^2(\epsilon) \]  \hspace{1cm} (11)

The expected value in (11) is directly dependent on the timing offset and decreases with increasing timing error. If it is lower than the certain threshold, it means that the timing error is large and the channel reliability factor should be decreased accordingly. This way, we make sure not to overestimate the \(L_c\) too much and keep the system very simple and robust.

Figure 3: The channel reliability factor’s offset (in dB) calculated from equation (10) for different timing offsets.

Figure 4: Performance of: a) turbo code with a given timing offset standard deviation (dashed line), b) turbo code according to the model (solid line) and c) turbo code with perfect estimation of the channel reliability value (dot-dash line).
The scheme is graphically shown in figure 5. In our case, we chose the threshold corresponding to 2dB overestimation, which appears when the timing error is equal to $\pm 0.26\, T_S$. We do not change the threshold depending on the SNR region - this will be discussed later in the paper.

**ALGORITHM MODIFICATION FOR PHASE ERRORS**

Large phase errors usually cause very significant loss in performance, which suggests that there is a large potential for improvement of the decoder - one of the possible approaches is to eliminate large errors of the phase estimate by concentrating them in the vicinity of zero.

In the idealized case of only phase offset, the incoming samples can be modelled as

$$y_k = e^{j\phi}c_k + n_k$$  \hspace{1cm} (12)

In order to reduce the effects of large phase errors we suggest a scheme that estimates the absolute phase error by the expected value of the absolute phase error.

If we assume that we know the probability distribution of the phase synchronizer (either by means of estimation or from its theoretical analysis) the expected value of the phase error is given by

$$\overline{\phi} = \int \phi |\rho_\phi(\phi)| d\phi$$  \hspace{1cm} (13)

which for Gaussian distribution will yield

$$\overline{\phi} = \sqrt{\frac{2}{\pi}} \sigma_\phi$$  \hspace{1cm} (14)

If we now use this expected value to correct the signal’s phase, all large phase offsets will be reduced at the expense of small phase offsets. In other words, by using our technique we modify the Gaussian distribution of the phase error to some other distribution with error values concentrated mostly in region of $[-\overline{\phi}, \overline{\phi}]$.

There remains, however, an unknown factor: the sign of the offset. One of the ways to circumvent this problem is to generate two sets of samples $y_1^k$ and $y_2^k$ as

$$y_{k}^{2} = y_{k}e^{j\overline{\phi}}$$  \hspace{1cm} (15)

and run the decoder for both sets. Since an error in the sign of the phase offset will result in reduced power of the phase corrected signal, the soft bits output of the MAP decoder should have smaller power than the soft bits output for the case of the correct sign (see [8]).

After running the MAP decoding for both sets, comparing the output power of the generated soft bits (after one iteration) and choosing the largest, we should be able to estimate the phase error sign and, consequently use the correct set of samples for remaining decoding iterations. The graphical illustration of the structure of the improved MAP decoder is shown in the figure 6.

By using the proposed algorithm we introduce only one additional iteration, since the results of the first decoding loop in the path with the correct sign can be used for consequent runs of the MAP algorithm. For 10 decoding iterations this corresponds to only 10% loss in delay which should be acceptable in most of the cases.

**PERFORMANCE**

The simulations were conducted with the (031,027) half-rate turbo code and an interleaver size of $N=256$. The number of decoding iterations was set to 10. The signal was transmitted on an AWGN channel, with QPSK modulation and root-raised cosine (with rolloff $\alpha = 0.3$) pulse shaping.

All BER calculations were done using data from 10000 transmitted blocks with independent, randomly generated parameters.
Figure 7 presents the bit error probability using the proposed scheme for the timing error. It can be seen that a simple thresholding algorithm has similar performance as the perfect estimator in the larger SNR region. Unfortunately, with decreasing signal power, the algorithm starts having problems since the threshold changes with changing Gaussian noise power. The obvious solution is to tailor the threshold to the expected variance of the Gaussian noise, which would complicate the algorithm. In the operating point region, however, the algorithm gives a gain of approximately 0.4dB.

Figure 8 presents the behaviour of the proposed phase offset algorithm. It is clear that our algorithm improves the performance of the traditional scheme, introducing a gain of approximately 0.5dB (which corresponds to 5 times lower BER).

**CONCLUSIONS**

It is obvious that the analysis of the performance of turbo codes must include the synchronization issues since the loss incurred can be quite significant, making convolutional codes an option, especially as they are not as sensitive to synchronization errors.

It is, however, relatively easy to cope with some effects of missynchronization, as we showed in this paper. We successfully increased the performance of a code by 0.4-0.5dB at the operating point, without using very sophisticated algorithms or major modifications of the classical BCJR algorithm. We achieved that without introducing any additional pilot symbols which saves bandwidth. Also the power of the signal remained the same which means that the power consumption will be only marginally increased.

It is quite probable that using more advanced algorithms could bring additional gains at the expense of hardware complexity. We believe, however, that our solution offers a good trade-off between performance and complexity of the decoder.

**REFERENCES**