NON-ITERATIVE JOINT CHANNEL EQUALISATION AND CHANNEL DECODING

A. Knickenberg, B.-L. Yeap, J. Hámorský, M. Breiling, L. Hanzo

Dept. of Electr. and Comp. Sc., Univ. of Southampton, SO17 1BJ, UK.
Tel: +44-703-593 125, Fax: +44-703-593 045
Email: lh@ecs.soton.ac.uk
http://www-mobile.ecs.soton.ac.uk

ABSTRACT

A non-iterative turbo equaliser scheme is proposed, which outperforms the iterative turbo equaliser by about 0.7 dB at a BER of 10^{-3} over a symbol-spaced two-path channel and by about 3.4 dB at a BER of 10^{-3} over a five-path Gaussian channel.

1. ITERATIVE TURBO EQUALISATION

In the context of iterative turbo equalisation [1, 4] both the convolutional encoder and the channel can be described with the aid of a trellis or a finite state machine, which are effectively concatenated and the maximum likelihood trellis path can be identified in each, for example using the Viterbi algorithm or Bahl's MAP algorithm [3]. This iterative algorithm has also been employed for the decoding of serially concatenated codes [2]. It is anticipated, however, that an amalgamated scheme achieves a better performance, than two separate algorithms.

The joint detection scheme, namely the turbo equalisation scheme of Reference [4] is portrayed in Figure 1. The scheme is based on a Maximum Aposteriori (MAP) equaliser, followed by a MAP decoder, which exchange in each iteration *a-priori* information about the encoded bits.

Before we elaborate further on the structure of Figure 1, we define the log-likelihood ratio (LLR), as the natural logarithm of the quotient of the probability that the bit d_k was a '1' and the probability that it was a '0':

$$LLR(d_k) = \ln \frac{P(d_k = 1)}{P(d_k = 0)} \tag{1}$$

This representation allows the hard decision concerning the estimated bits to be conveniently carried out at the end of the iterations with respect to the threshold of zero.

The MAP equaliser and decoder blocks are so-called soft-in soft-out blocks, which allow them to readily exchange 'soft'-information during each iteration. The MAP equaliser of Figure 1 is fed with the channel output values $\vec{y}$ and with the a-priori information $L_e^D(\hat{c}')$ concerning the encoded bits $\hat{c}'$. As expected, in the first iteration we have $L_e^D(\hat{c}') = 0$. Upon invoking the MAP algorithm, the equaliser of Figure 1 outputs the *a-posteriori* LLR values $L^E(\hat{c}')$ related to the encoded bits, where the *a-posteriori* LLR is composed of the channel information, the a-priori information plus the extrinsic information. However, in order to generate the decoded bits, we want to pass to the deinterleaver and decoder block of Figure 1 only the channel information and the extrinsic information, where the latter is defined as the information concerning a certain bit $c_{k'}$, which is calculated by considering the received sequence upon explicitly excluding the values due to the bit $c_{k'}$ - hence the term extrinsic.

Therefore, in order to remove the influence of the a-priori information, it is subtracted from $L^E(\hat{c}')$, yielding the sum of channel information and extrinsic information, namely $L_*^E(\hat{c}')$, which is formulated as:

$$L_*^E(\hat{c}') = L^E(\hat{c}') - L_e^D(\hat{c}').$$

These values $L_*^E(\hat{c}')$ are then deinterleaved in Figure 1, in order to be in the right order for feeding them into the MAP decoder, as a-priori information.

The MAP decoder then delivers the a-posteriori values of the encoded bits $L^D(\hat{c}')$. The LLR values of the data bits are then subjected to hard decision following the last iteration. Now again, we have to subtract from the LLR values of the encoded bits the quantity $L_*^E(\hat{c})$, in order to acquire the extrinsic information $L_e^D(\hat{c})$ concerning the encoded bits, which is given by:

$$L_e^D(\hat{c}) = L^D(\hat{c}) - L_*^E(\hat{c}).$$

After passing through the interleaver of Figure 1, the extrinsic information of the decoder is ready to be used as a-priori information by the equaliser for the next iteration. By feeding back the extrinsic information iteratively, we will get an improved BER from iteration to iteration. Let us now consider a novel technique, which we refer to as non-iterative joint equalisation/decoding.

In summary, in iterative turbo equalisation and decoding there are two consecutive trellises, which exchange soft-decision information concerning the encoded bits, passing the associated log-likelihood values back and forth a number of times, in order to iteratively improve the system's bit error rate (BER) performance. The maximum likelihood information sequence can be asymptotically approached by

The authors are with Department of Electronics and Computer Sciences, University of Southampton, SO17 1BJ, U.K. E-mail: lh@ecs.soton.ac.uk, http://www-mobile.ecs.soton.ac.uk.

The financial support of the following organisations is gratefully acknowledged: Motorola ECID, Swindon, UK; European Community, Brussels, Belgium; Engineering and Physical Sciences Research Council, Swindon, UK; Mobile Virtual Centre of Excellence, UK.; The Royal Society (NATO Fellowship).

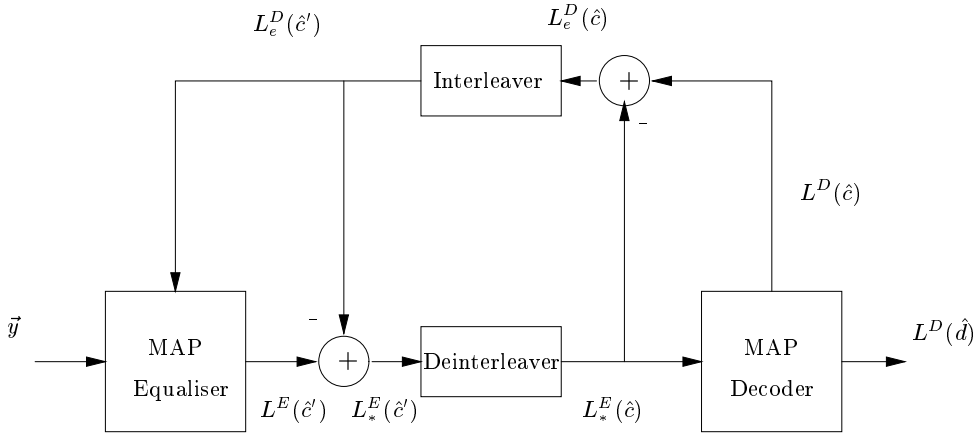

Figure 1: The principle of iterative turbo equalisation of Reference [4] .

this method upon increasing the number of iterations. Let us now consider a non-iterative technique in the next section.

2. NON-ITERATIVE TURBO EQUALISATION

In Reference [5] the optimum non-iterative turbo-decoding algorithm was introduced for the decoding of parallel concatenated codes, which are also referred to as turbo codes. **By contrast, in this contribution we propose a non-iterative joint channel decoding and equalisation scheme for the serially concatenated convolutional encoder plus dispersive channel arrangement of Figure 2, which outperforms iterative turbo equalisers**. In contrast to iterative turbo equalisation algorithms [1, 4], where we need a certain number of iterations, in order to achieve a good performance, according to our proposed solution we carry out equalisation and decoding in a joint, non-iterative step. We note here that conceptually a multipath channel can be seen as a special convolutional encoder, having a coding rate of $R = 1$ and a memory length $K - 1$, which is equal to the maximum delay of the channel expressed in terms of the time units k' characteristic of the output of the decoder. We then have a system of two serial encoders, namely the convolutional encoder and the channel, separated by an interleaver, namely the channel interleaver.

For the proposed non-iterative joint equalisation/decoding process we have to construct a finite-state machine, which models the whole system. The input of this finite-state machine is constituted by the the original data bits and its output is the channel output. In a finite-state machine the output $\vec{y_k}$ and the next state S_{k+1} are fully determined by the current state S_k and the current input data bit d_k. However, the interleaver renders the receiver's task complex.

Let us consider the example of Figure 2, where the multipath channel has $L = 2$ symbol-spaced paths, corresponding to an inner encoder memory length of $\nu_I = L - 1 = 1$. The convolutional outer encoder of Figure 2 has a constraint length of $K = 3$ and a memory length of $\nu_o = K - 1 = 2$. Each encoded bit is determined by the current incoming data bit d_k and the ν_o previous ones. These encoded bits pass through the interleaver and enter the multipath channel in the interleaved order. We denote the interleaver function by $l(k')$, which assigns the input time instant k' to the output time instant $l(k')$. Hence the output of the channel at the time instant k' depends on $c_{l(k')}, c_{l(k'-1)}, ..., c_{l(k'-\nu_I)}$. Each convolutionally encoded bit in turn depends on the last $\nu_o + 1$ data bits. Finally, we get the output of the channel as a function of the data bits as follows:

$$\begin{aligned} y_{k'} &= g_0 \left[d_{l(k)} \oplus d_{l(k)-1} \oplus \cdots \oplus d_{l(k)-\nu_o}\right] + \\ &+ g_1 \left[d_{l(k-1)} \oplus d_{l(k-1)-1} \oplus \cdots \oplus d_{l(k-1)-\nu_o}\right] + \\ &\vdots \\ &+ g_{\nu_I} \left[d_{l(k-\nu_I)} \oplus d_{l(k-\nu_I)-1} \oplus \cdots \oplus d_{l(k-\nu_I)-\nu_o}\right], \end{aligned}$$

where g_i represents the symbol-spaced dispersive channel impulse response taps. In conventional trellises, which are used to describe convolutional codes, the encoder state is determined by the bits in the encoder's shift-register. In our serially concatenated convolutional encoder plus channel **we define a super-state S_k^*, which is determined by all the bits that are used to calculate the channel output, except the actual input bit**, which is often referred to as the state transition bit.

In our forthcoming discourse we propose a non-iterative equalisation / decoding algorithm and highlight its operation using the simple example of a $2 \times N$ block interleaver and a half-rate convolutional encoder.

2.1. Non-iterative joint equalisation/decoding using a $2 \times N$ interleaver

Again, Figure 2 shows a simple manifestation of the serially concatenated system using a non-recursive, half-rate convolutional encoder with a constraint length of $K = 3$, octal generator polynomials $G[0] = 5$ and $G[1] = 7$, a $2 \times N$ block interleaver and a two-path channel. It is readily seen that this two-column interleaver will simply separate for each encoded bit block of length $2N$ the even and the odd encoded bits. First all the odd indexed encoded bits generated by the first generator polynomial will enter the channel, followed by the even indexed encoded bits due to the second

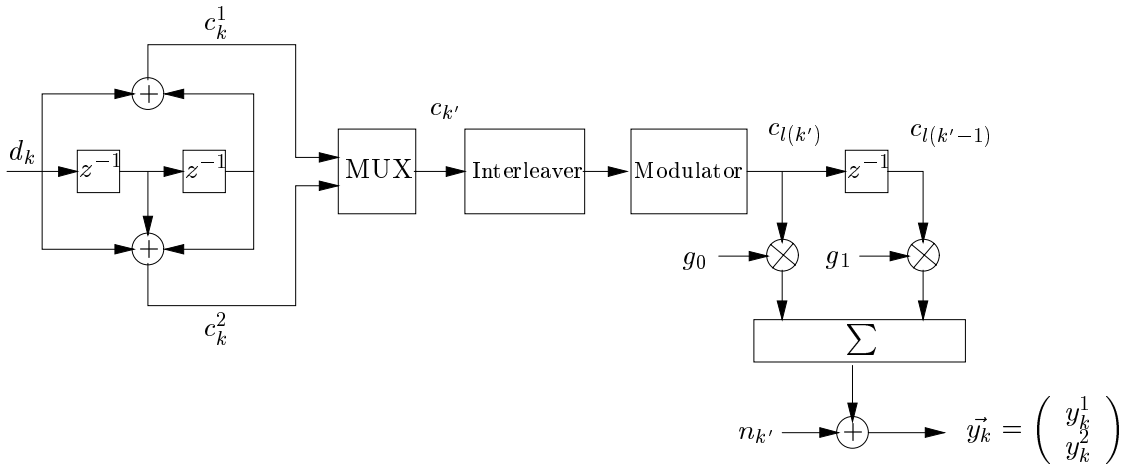

Figure 2: Model of the transmitter consisting of the convolutional encoder, interleaver, modulator and complex channel.

generator polynomial. The super state S_k^* is constituted by all the previous bits, which influence the channel output sequence, namely y_k^1 and y_k^2 in Figure 2. Let us denote the encoded bits emanating from the first generator polynomial at the time k by c_k^1 and those due to the second one by c_k^2. The channel outputs y_k^1 and y_k^2 in Figure 2 are calculated according to the following equations:

$$\begin{aligned} y_k^1 &= g_0 \cdot c_k^1 + g_1 \cdot c_{k-1}^1 \qquad (2) \\ &= g_0 \cdot (d_k \oplus d_{k-2}) + g_1 \cdot (d_{k-1} \oplus d_{k-3}) \\ y_k^2 &= g_0 \cdot c_k^2 + g_1 \cdot c_{k-1}^2 \qquad (3) \\ &= g_0 \cdot (d_k \oplus d_{k-1} \oplus d_{k-2}) + g_1 \cdot (d_{k-1} \oplus d_{k-2} \oplus d_{k-3}), \end{aligned}$$

where y_k^1 is transmitted during the first half of the encoded output block of length $2N$ and y_k^2 during the second half. The superstate S_k^* is determined by the data bits $(d_{k-1}, ..., d_{k-3})$, while d_k is the current encoder input bit. The next superstate, namely S_{k+1}^* will be determined by $(d_k, ..., d_{k-2})$.

In our example the outer encoder's and the inner encoder's memory length are given by $\nu_o = 2$ and $\nu_I = 1$, respectively, hence the number of superstate bits is $\nu_o + \nu_I$ and the number of superstates is $2^{\nu_o+\nu_I}$. With each additional memory-element in the channel, we have to consider one additional previous encoded bit, implying that we have to consider an additional bit in the superstate.

Let us now consider the inner-working of the proposed non-iterative scheme at the commencement of operation and also near the middle of the transmitted sequence, where we change from transmitting c_k^1 to c_k^2, and lastly, at the end of each encoded block of length $2N$. Let us assume for the sake of illustration that we have a 2×10 interleaver. Then we have 20 encoded bits and hence 10 input data bits at the encoder. In order to be in a pre-defined state at the end of each encoded block we will use a terminated code with 3 zero-valued termination bits. The length of this termination sequence is equal to the number of concatenated encoder state bits i.e. the number of bits in the superstate. This implies that in the 10-bit encoded sequence we have 7 data bits and 3 termination bits, which are, again, logical '0's, since we are using a non-recursive encoder. Table 1 shows the situation at the above-mentioned 'critical' operational points of the encoder. Observe in the example that if the input bits $d_8, ..., d_{10}$ are zero, then c_{10}^1 and c_{10}^2 are also zero.

Time k'	State	$c_{l(k')}$	$c_{l(k'-1)}$	Output y_k^i
1	$(0,0,0)$	$c_1^1 = d_1 \oplus 0$	0	y_1^1
2	$(d_1,0,0)$	$c_2^1 = d_2 \oplus 0$	$c_1^1 = d_1 \oplus 0$	y_2^1
⋮	⋮	⋮	⋮	⋮
10	(d_9, d_8, d_7)	$c_{10}^1 = d_{10} \oplus d_8$ $=0$	$c_9^1 = d_9 \oplus d_7$	y_{10}^1
11	$(0,0,0)$	$c_1^2 = d_1 \oplus 0 \oplus 0$	$c_{10}^1 = d_{10} \oplus d_8$ $=0$	y_1^2
12	$(d_1,0,0)$	$c_2^2 = d_2 \oplus d_1 \oplus 0$	$c_1^2 = d_1 \oplus 0 \oplus 0$	y_2^2
⋮	⋮	⋮	⋮	⋮
20	(d_9, d_8, d_7)	$c_{10}^2 = d_{10} \oplus d_9 \oplus d_8$ $=0$	$c_9^2 = d_9 \oplus d_8 \oplus d_7$	y_{10}^2

Table 1: The contents of the concatenated encoder seen in Figure 2 versus time k'.

The proposed decoder operates on the basis of constructing the encoder's super-trellis constituted by the previously defined super-states S_k^* described by the data bits $(d_{k-1}, ..., d_{k-3})$ and invoking the Log-MAP algorithm for joint channel equalisation and decoding. Although in this context also the Viterbi algorithm could have been utilized, since in this particular application there is no need for producing the soft outputs provided by the MAP algorithm. More explicitly, the proposed non-iterative scheme refrains from passing soft-decision information iteratively between the separate trellises of the equaliser and channel decoder and operates on the basis of the super-trellis, rather than on the two independent trellises of the equaliser and channel decoder. Invoking the supertrellis in a non-iterative step improves the performance of the amalgamated scheme.

3. SYSTEM PERFORMANCE

We have conducted simulations using the proposed non-iterative algorithm described above and compared its performance to that of the iterative turbo equalisation algorithm using the same parameters. In both systems we employed BPSK modulation.

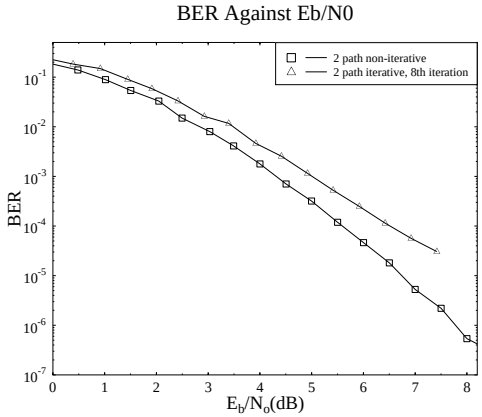

Figure 3: Comparison of iterative versus non-iterative joint equalisation/decoding over a two-path dispersive Gaussian channel, when using a convolutional code having the following parameters: $R = \frac{1}{2}$, $K = 5$, $G[0]$=37, $G[1]$=21, channel impulse response taps of $g_0 = g_1 = \frac{1}{\sqrt{2}}$ and an interleaver of 2×2052 bits

In Figure 3 we have compared the iterative and non-iterative algorithms, when transmitting over a dispersive two-path Gaussian channel using eight iterations. We used a non-recursive rate $R = \frac{1}{2}$, constraint length $K = 5$ convolutional code with a generator polynomial of $G[0] = 37$ and $G[1] = 21$, which are expressed in octal format. The channel impulse response taps were given by $g_0 = g_1 = \frac{1}{\sqrt{2}}$ and the interleaver size was 2×2052=4104. As seen in the figure, the non-iterative algorithm out-performs the iterative scheme by about 0.7 dB at a BER of 10^{-3}.

In order to further quantify the proposed scheme's performance, in Figure 4 we compared the two algorithms over a more dispersive five-path channel, employing the following parameters: $R = \frac{1}{2}$; constraint length of $K = 5$; $G[0] = 37$ and $G[1] = 21$; channel coefficients of $g_0 = g_4 = 0.227$, $g_1 = g_3 = 0.46$, $g_2 = 0.688$; interleaver size of 2×2052=4104; eight iterations. Over this more dispersive channel apparently the non-iterative scheme's performance becomes even more attractive, resulting in an SNR-gain of ≈ 3.4 dB at a BER of 10^{-3}. We also note that the performance of the Max-Log-MAP and the Log-MAP algorithms was also compared and was found to be virtually identical. These performance improvements were achieved in the context of the $2 \times N$ interleaver at the cost of a reasonable complexity. For example, in the case of $K = 5$ and a two-path channel we have $\nu_o + \nu_I = 4 + 1$ state bits, hence there are $2^5 = 32$ states. Even for $K = 5$ and a five-path channel we have a moderate number of $2^8 = 256$ states.

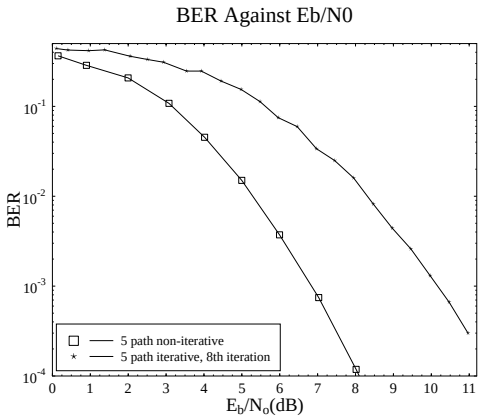

Figure 4: Comparison of iterative versus non-iterative turbo equalisation over a five-path Gaussian channel using a convolutional code having the following parameters: $R = \frac{1}{2}$, K=5, $G[0] = 37$, $G[1] = 21$, $g_0 = g_4 = 0.227$, $g_1 = g_3 = 0.46$, $g_2 = 0.688$, 2×2052 interleaver.

3.1. Effect of Interleaver Depth

Let us now consider the effect of interleaver-depth in this section. If we do not use an interleaver, then the encoded bits enter the channel in the same order as they were calculated by the encoder. We then have two channel output bits, namely y_k^1 and y_k^2, which were due to the input data bit d_k, when the super-state was S_k^*. If we consider the example of Figure 2, we have the following equations:

$$\begin{aligned} y_k^1 &= g_0 \cdot c_k^1 + g_1 \cdot c_{k-1}^2 \qquad (4) \\ &= g_0(d_k \oplus d_{k-2}) + g_1(d_{k-1} \oplus d_{k-2} \oplus d_{k-3}) \\ y_k^2 &= g_0 \cdot c_k^2 + g_1 \cdot c_k^1 \qquad (5) \\ &= g_0(d_k \oplus d_{k-1} \oplus d_{k-2}) + g_1(d_k \oplus d_{k-2}). \end{aligned}$$

Hence we have three state bits in the superstate $S_k^* = (d_{k-1}, d_{k-2}, d_{k-3})$. Due to the lack of interleaving we have now $\nu_o + \lfloor \nu_I/2 \rfloor$ state bits, which implies that we need less state bits, than the previous $\nu_o + \nu_I$ value, which was associated with the $2 \times N$ interleaver.

When using a $4 \times N$ interleaver, we have to cope with a higher detection complexity. Specifically, this interleaver produces from the sequence:

$$..., c_4^2, c_4^1, c_3^2, c_3^1, c_2^2, c_2^1, c_1^2, c_1^1 \Longrightarrow$$

the following output sequence:

$$..., c_6^2, c_4^2, c_2^2, ..., c_6^1, c_4^1, c_2^1, ..., c_5^2, c_3^2, c_1^2, ..., c_5^1, c_3^1, c_1^1.$$

In the shift-register stages of Figure 2 modelling the channel we now have the following encoded bits: $c_k^i, c_{k-2}^i, ...$ Hence, in comparison to the $2 \times N$ interleaver we now always have a displacement of two clock intervals between the indices of the coded bits, which consecutively enter the channel. If we consider again the example of Figure 2, then the two

output values are calculated as follows:

$$y_k^1 = g_0 \cdot c_k^1 + g_1 \cdot c_{k-2}^1 \quad (6)$$
$$= g_0(d_k \oplus d_{k-2}) + g_1(d_{k-2} \oplus d_{k-4})$$
$$y_k^2 = g_0 \cdot c_k^2 + g_1 \cdot c_{k-2}^2 \quad (7)$$
$$= g_0(d_k \oplus d_{k-1} \oplus d_{k-2}) + g_1(d_{k-2} \oplus d_{k-3} \oplus d_{k-4}).$$

This leads to the following superstate:

$$S_k^* = (d_{k-1}, d_{k-2}, d_{k-3}, d_{k-4}).$$

This example shows that we now need $\nu_o + 2 \cdot \nu_I$ state bits, since we have to store all the previous bits, which are either used to calculate the channel output or which are used to determine the next state. In this case the number of states increases rapidly with the memory or dispersion of the channel. For example, for a five-path channel and a convolutional encoder of constraint length $K = 5$ this would result in $2^{4+2\cdot4} = 2^{12} = 4096$ states, which is irrealistically high.

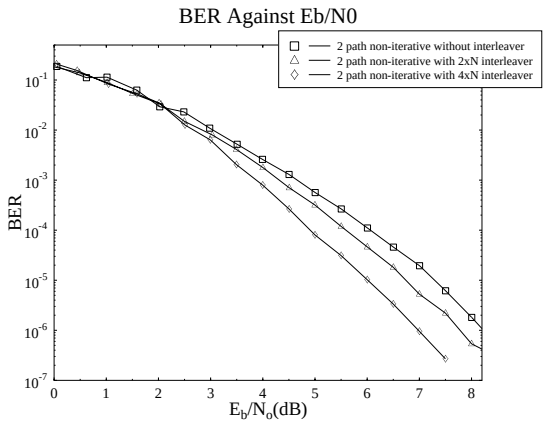

Figure 5: Comparison of different interleavers, using non-iterative joint equalisation/decoding over a 2 path channel. $R = \frac{1}{2}$, K=5, $G[0]$=37, $G[1]$=21, $g_0 = g_1 = \frac{1}{\sqrt{2}}$.

We have conducted simulations for a two-path channel using the same parameters, as in Section 2.1, except for the interleaver. Figure 5 shows the results for the two-path channel without interleaver, with a $2 \times N$ or a $4 \times N$ interleaver.

4. CONCLUSIONS

The proposed non-iterative joint equaliser/decoding outperformed the iterative turbo equaliser by about 0.7 dB at a BER of 10^{-3} over the two-path channel and by about 3.4 dB at a BER of 10^{-3} over the five-path Gaussian channel.

5. REFERENCES

[1] C.Douillard, A.Picart, M.Jézéquel, P.Didier, C.Berrou, and A.Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *European Transactions on Communications*, vol. 6, pp. 507–511, September-October 1995.

[2] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, "Serial Concatenation of Interleaved Codes: Perfomance Analysis, Design, and Iterative Decoding," *TDA Progress Report*, pp. 42–126, August 1996.

[3] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimising Symbol Error Rate," *IEEE Transactions on Information Theory*, pp. 284–287, March 1974.

[4] G.Bauch, H.Khorram, and J.Hagenauer, "Iterative Equalization and Decoding in Mobile Communications Systems," in *European Personal Mobile Communications Conference*, pp. 301–312, 1997.

[5] M. Breiling, L. Hanzo: Non-iterative Optimum Supertrellis Decoding of Turbo Codes, IEE Electr. Letters, 8th May, 1997, Vol. 33, No. 10, pp 848-849