BULETINUL INSTITUTULUI POLITEHNIC DIN IAȘI
Publicat de
Universitatea Tehnică „Gheorghe Asachi" din Iaşi
Tomul LVIII (LXII), Fasc. 1, 2012
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ

ABOUT SYMMETRIC CONVOLUTIONAL CODES OF MEMORY TWO AND THREE USED IN SPACE-TIME TURBO CODING

BY

ANA-MIRELA ROTOPĂNESCU* and LUCIAN TRIFINA

"Gheorghe Asachi" Technical University of Iaşi
Faculty of Electronics, Telecommunications and Information Technology

Received: December 9, 2011
Accepted for publication: February 19, 2012

Abstract. In this paper are analysed the performances of different symmetric convolutional turbo codes for a transmission system that uses a doubly iterative decoding algorithm, with multiple antenna diversity. The turbo encoder uses an Ω'–QPP (Quadratic–Permutation–Polynomial) interleaver, and two different decoding algorithms, namely, the Max–Log–APP (Maximum–Logarithm–A–Posteriori–Probability) algorithm and second, the Log–APP (Logarithm–A–Posteriori–Probability) algorithm. Also, it was proposed an increase of the memory of the encoder, from order 2 to 3, to highlight the improvement of the FER (Frame–Error–Rate) and BER (Bit–Error–Rate) performances.

Key words: space-time turbo codes; doubly iterative decoder; BER/FER performances.

1. Introduction

There were analysed the FER (Frame–Error–Rate) and BER (Bit–Error–Rate) performances obtained for a transmission system of reduced complexity that uses a doubly iterative decoding algorithm, where the

*Corresponding author: *e-mail*: rotopanescu.mirela@yahoo.com

component turbo decoder presents an interface to eliminate the space interference (Biglieri *et al*., 2005). Previously it was shown that for turbo codes, the Ω'–QPP (Quadratic–Permutation–Polynomial) interleaver used in this scheme brings an improvement in the performances compared to those obtained when using a random interleaver (Rotopănescu & Trifina, 2011). Also, it was shown that the Log–APP (Logarithm–A–Posteriori–Probability) algorithm brings performance improvements compared to the Max–Log–APP (Maximum–Logarithm–A–Posteriori–Probability) algorithm (Rotopănescu & Trifina, 2011).

Baltă & Kovaci (2004) have made a performance analysis of the convolutional codes used in different forms of concatenation of classical turbo codes, for different memory orders and for different decoding algorithms. Therefore, in this paper is made an analysis of several combinations of convolutional codes used in turbo codes for the proposed system.

Further, it is presented the system model, then are described the symmetric convolutional codes chosen for simulations, and finally are given the simulation results and the conclusions of the paper.

2. System Model

It was considered a mobile communication system with 16 transmission antennas and 16 receiver antennas, giving a spectral efficiency of 16 bits/s/Hz. The information bits are turbo-coded and the interleaved bits that give the encoded vector are serial to parallel converted and mapped into a signal constellation. The signals at the modulator output are transmitted by each antenna, at each time instant.

The receiver uses a linear MMSE (Minimum–Mean–Square–Error) interface. This consists in a linear filter that minimizes the root mean square error. The additional interleaver is used to remove the correlation between the transmitted consecutive bits, helping in the decoding process. Its size is chosen in such a manner that there is no further increase in the system delay. Also, this is necessary to decorrelate the LLR (Log–Likelyhood–Ratio) of the adjacent bits.

The transmitter and the receiver block schemes are given Figs. 1 and 2.

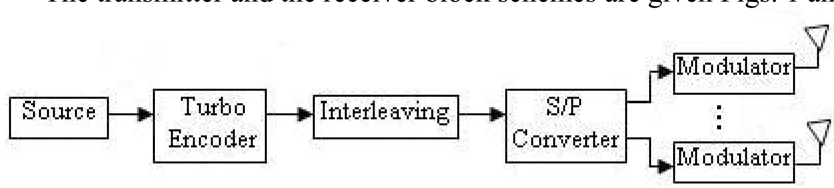

Fig. 1 – Transmitter block scheme.

The simulations were performed using QPSK (Quadrature–Phase–Shift–Keying) modulation (M = 2), for a memory order equal to 2 and then equal to 3.

The symmetric turbo codes have the global coding rate 1/2 for memory 2 and 1/3 for memory 3. We use forward and feedback generating polynomials written in octal form. The space-time code word is a matrix having 130 columns.

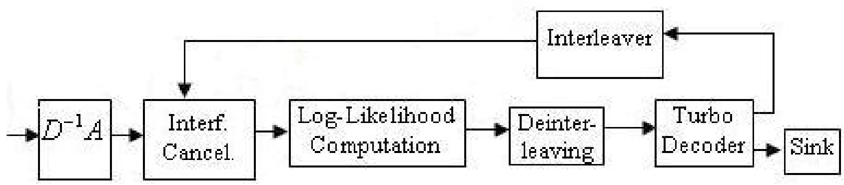

Fig. 2 – Receiver block scheme.

The turbo encoder and decoder uses a Ω'–QPP interleaver of length 2080, because this leads to an improved performance for large lengths. The free term is found by maximizing the corner merit because only the first trellis is terminated, and the margin effects of trellis termination are avoided. Between the turbo encoder and the serial to parallel converter will be used a random interleaver.

The decoder will use either the Max–Log–APP decoding algorithm with extrinsic information scaling (Trifina *et al.*, 2011) or the Log–APP algorithm (Rotopănescu & Trifina, 2011). In the analysis when the encoder memory is 2, we use both decoding algorithms, Max–Log–Log–APP and Log–APP, and if memory is 3, we use only the Log–APP algorithm.

To eliminate the spatial interference a number of k = 0, k = 1 or k = 4 iterations was used. For the Max–Log–APP decoding algorithm with extrinsic information scaling, for k = 0 we consider the scaling factor s = 0.9, for k = 1, s = 0.8 and for k = 4, s = 0.75 (Trifina *et al.*, 2011).

For both cases, when using the Log–APP algorithm, and the Max–Log–APP algorithm, the turbo decoder performs a number of 10 iterations with Genie-stopper stopping criterion, meaning that the iterations in turbo decoding are stopped when the decoded bit frame is identical to the information bit frame originally coded. Also, the number of distinct blocks with constant fading will be considered equal to 1.

The value of the signal to noise ratio for which the simulations were performed was chosen as the largest assumed in simulations (Biglieri *et al.*, 2005). Thus, for memory 2, for both Max–Log–APP and Log–APP algorithms, for k = 0, the SNR was considered equal to –5 dB, and for k = 1 and k = 4, the SNR is –6.5 dB.

For the simulations for which the encoder memory is considered 3, we use only the Log–APP algorithm. In this case there were performed simulations for k = 0 to a value of SNR = –5 dB, for a number of blocks, noted in tables with *nb* equal to 2,000,000, and we did not find any errors. Thus, it was decided a smaller value for signal to noise ratio so for $k = 0$ the value at which simulations were performed is SNR = –6 dB. For $k = 1$, SNR = –6.5 dB, and for $k = 4$ simulation results were not given because they are degraded compared to those obtained in the analysis with Log–APP algorithm for memory 2, due to the relative errors introduced by the doubly iterations .

3. Symmetric Turbo Codes for Antenna Diversity Systems

The polynomial pairs given in octal form are denoted by p_1/p_2 , where p_1 is the forward polynomial and p_2 – the feedback polynomial. We assume all combinations of bits to form the forward polynomials. The feedback polynomials are chosen so that the first and last bit are equal to 1 for the feedback in the recursive systematic encoder. Thus, for memory 2, we will choose the feedback polynomials 5 (101) and 7 (111) and for memory 3 we choose the feedback polynomials 11 (1001), 13 (1011), 15 (1101) and 17 (1111).

In the tables are given the number of errors, denoted by *er*, the number of erroneous blocks, denoted by *nber* and the number of transmitted blocks noted with *nb*.

The BER value is obtained by dividing the number of errors after decoding to the number of transmitted bits, where the total number of transmitted bits is equal to the number of transmitted blocks multiplied by the information frame length. The information frame length is 2,078, for memory 2 and 2,077, for memory 3. The information frame length will be considered 2,077 or 2,078 because the interleaver length is 2,080 and only the first trellis is finished, and the second is not. In this method a number of bits is added to the end of the frame equal to the encoder memory order. Obviously these bits are not information ones and must be subtracted (here 2,080 – 2 = 2,078 and 2,080 – 3 = 2,077). The FER value, both when the memory is 2 as well as when the memory is 3, is obtained by dividing the number of erroneous blocks to the number of transmitted blocks.

4. Component Codes Searching from Simulation Results

In Tables 1 and 2 are given the BER and FER performances of various component convolutional codes, for the case when the memory of the encoder is 2, the decoding algorithm is Max–Log–APP, and the number of doubly iterations is equal to $k = 0$, $k = 1$ and $k = 4$. In Tables 3 and 4 are given, respectively, the same performances as in the Tables 1 and 2, except that the decoding algorithm used is Log–APP. In Table 5 are given the BER and FER performances obtained in the case when the encoder memory is 3, for a number of doubly iterations equal to k = 0 and k = 1.

Table 1
Performances for Max–Log–APP Algorithm for k=0 and k=1

	k = 0, SNR = –5 dB					k = 1, SNR = –6.5 dB				
p_1/p_2	BER	FER	*er*	*nber*	*nb*	BER	FER	*er*	*nber*	*nb*
1/5	*2.29e-2*	*1.00e+0*	*47,688,979*	*1,000,000*	*1,000,000*	*3.90e-2*	*1.00e+0*	*430,520*	*5,300*	*5,300*
1/7	2.05e-5	5.21e-3	42,685	5,218	1,000,000	4.42e-4	5.88e-2	92,043	5,887	100,000
2/5	2.29e-2	1.00e+0	47,666,763	1,000,000	1,000,000	3.66e-2	1.00e+0	7,616,152	100,000	100,000
2/7	2.01e-5	4.96e-3	41,933	4,965	1,000,000	4.48e-4	5.623e-1	369,305	22,287	396,000
3/5	3.96e-3	5.65e-1	6,797,396	466,547	824,500	1.20e-2	9.17e-1	2,038,762	74,655	81,400
3/7	1.22e-4	2.35e-2	254,561	23,516	1,000,000	1.50e-3	1.97e-1	3,124,205	197,111	1,000,000
4/5	2.28e-2	1.00e+0	455,518	9,600	9,600	2.66e-2	1.00e+0	210,563	3,800	3,800
4/7	2.04e-5	5.09e-3	42,455	5,092	1,000,000	1.98e-5	3.52e-3	41,277	3,527	1,000,000
5/7	**1.08e-7**	**3.80e-5**	**225**	**38**	**1,000,000**	**3.90e-6**	**1.24e-4**	**8,116**	**124**	**1,000,000**
6/5	3.94e-3	5.61e-1	8,201,604	561,651	1,000,000	3.51e-3	5.49e-1	730,049	5,4935	100,000
6/7	1.22e-4	2.33e-2	254,587	23,399	1,000,000	1.22e-4	2.54e-2	81,514	8,155	320,400
7/5	4.62e-3	9.69e-1	7,410,426	747,813	771,500	3.59e-3	9.39e-1	746,663	93,948	100,000

Table 2
Performances for Max–Log–APP Algorithm for k=4

	k = 4, SNR = –6.5 dB				
p_1/p_2	BER	FER	*er*	*nber*	*nb*
1/5	*4.04e-2*	*1.00e+0*	*5,484,280*	*65,300*	*65,300*
1/7	2.13e-4	4.46e-2	29,262	2,951	66,100
2/5	3.77e-2	1.00e+0	6,100,071	77,800	77,800
2/7	2.14e-4	4.30e-2	394,802	38,023	884,200
3/5	1.12e-2	9.13e-1	21,164,358	828,390	907,100
3/7	9.21e-4	2.07e-1	1,915,011	207,688	1,000,000
4/5	2.48e-3	1.00e+0	2,092,143	405,600	405,600
4/7	5.22e-7	2.39e-4	652	144	600,600
5/7	**1.95e-7**	**2.00e-6**	**407**	**2**	**1,000,000**
6/5	7.60e-4	2.32e-1	1,245,410	183,371	787,700
6/7	4.56e-6	3.81e-3	948	381	100,000
7/5	2.10e-3	8.67e-1	2,444,338	484,265	558,400

Table 3
Performances for Log–APP Algorithm for k = 0 and k = 1

	k = 0, SNR = –5 dB					k = 1, SNR = –6.5 dB				
p_1/p_2	BER	FER	*er*	*nber*	*nb*	BER	FER	*er*	*nber*	*nb*
1/5	*2.30e-2*	*1.00e+0*	*4,798*	*100*	*100*	*3.71e-2*	*1.00e+0*	*7,724*	*100*	*100*
1/7	1.64e-5	4.55e-3	736	98	21,500	2.93e-4	3.36e-2	1,814	100	2,976
2/5	2.25e-2	1.00e+0	4,694	100	100	3.63e-2	1.00e+0	3,990,706	52,800	52,800
2/7	1.93e-5	4.67e-3	870	101	21,600	3.55e-4	3.30e-2	69,149	3,094	93,600
3/5	3.66e-3	5.40e-1	11,419	810	1,500	1.00e-2	8.03e-1	68,583	2,650	3,300
3/7	1.12e-4	2.00e-2	586	50	2,500	1.06e-3	9.60e-2	17,208	749	7,800
4/5	2.25e-2	1.00e+0	4,693	100	100	2.61e-2	1.00e+0	4,676,183	86,200	86,200
4/7	1.98e-5	4.82e-3	1,317	154	31,900	9.37e-6	1.35e-3	7,995	557	410,600
5/7	**1.11e-7**	**3.99e-5**	**435**	**75**	**1,877,100**	**1.21e-6**	**4.60e-5**	**2,534**	**46**	**1,000,000**
6/5	3.43e-3	5.12e-1	2,854	205	400	2.36e-3	3.48e-1	491,999	34,817	100,000
6/7	1.00e-4	1.93e-2	897	83	4,300	5.02e-5	7.12e-3	5,592	381	53,500
7/5	4.46e-3	9.65e-1	665,588	69,211	71,700	3.13e-3	8.94e-1	331,740	45,601	51,000

Table 4

Performances for Log–APP Algorithm for k = 4

k = 4, SNR = –6.5 dB					
p_1/p_2	BER	FER	*er*	*nber*	*nb*
1/5	*3.85e-2*	*1.00e+0*	*208,089*	*2,600*	*2,600*
1/7	1.57e-4	1.41e-2	32,659	1,410	100,000
2/5	3.68e-2	1.00e+0	1,587,159	20,700	20,700
2/7	1.66e-4	1.37e-2	34,541	1,378	100,000
3/5	8.83e-3	7.01e-1	411,334	15,720	22,400
3/7	5.83e-4	5.06e-2	121,320	5,068	100,000
4/5	2.39e-2	1.00e+0	139,431	2,800	2,800
4/7	2.30e-7	9.00e-5	48	9	100,000
5/7	**9.76e-8**	**1.00e-6**	**203**	**1**	**1,000,000**
6/5	2.62e-4	6.63e-2	2,561	312	4,700
6/7	5.66e-7	1.73e-4	34	5	28,900
7/5	1.71e-3	7.62e-1	8,555	1,830	2,400

For the cases mentioned in Tables 1 and 2 the best performances are given by the code 5/7 (bold) and the worst performances are obtained for the code 1/5 (italic).

From these tables we see that the codes that have the feedback polynomials 7 in octal form lead to better FER and BER performances than those which have the feedback polynomial equal to 5. The codes that are denoted as $p_1/7$, can reach the order 10^{-5} in BER performances, while those in the form $p_1/5$ can reach a minimum order of 10^{-3}.

When the value of FER performance is equal to 1.00e+0, it means that the SNR value was chosen too big to calculate a real FER value. For each SNR value chosen for each k, we see that for the same code $p_1/7$, the FER performances are better than those obtained for $p_1/5$ codes. This way we see again the slow performances of the non-primitive feedback polynomials.

From Tables 1 and 2 we see that for the same value SNR = –6.5 dB, when the number of doubly iterations is $k = 4$, both FER and BER are better or, in some cases, are almost equal to those obtained when $k = 1$. For example, for the code 5/7, for $k = 1$, the BER value obtained is 3.90e–6, while for $k = 4$ the obtained BER value is 1.95e–7. For $k = 1$, the FER value is 1.24e–4, and for $k =$ = 4, is much better, 2.00e–6. A visible improvement can be seen at the code 4/7. The BER value obtained for $k = 1$ is 1.98e–5, and for $k = 4$, is 5.22e–7, and the FER value is 3.52e–3 for $k = 1$, and 2.39e–4 for $k = 4$.

From Tables 3 and 4 are drawn the same conclusions as in the Tables 1 and 2. In these tables, the performance values were obtained when the turbo decoder uses the Log–APP decoding algorithm. Comparing the obtained BER and FER performances, for codes p_1/p_2 , for the same value of k, we see that these values are better than those obtained in the case when the turbo decoder uses the Max–Log–APP algorithm.

For example, for code 5/7, using the Max–Log–APP algorithm, for $k = 0$, SNR = –5 dB, the obtained BER value is 1.08e–7, while using the Log–APP algorithm this value is 1.11e–7. Also, under the same conditions, the obtained FER value when the decoder is using the Max–Log–APP algorithm is

3.80e–5 and for the second algorithm is 3.99e–5. When k = 1, for the same code, was obtained a BER value equal to 3.90e–6 for the Max–Log–APP algorithm and BER = 1.21e–6 for the Log–APP algorithm. When k = 4, the BER value obtained for the Max–Log–APP algorithm is 1.95e–7, and for the Log–APP algorithm is 9.76e–8. Comparing the FER values can be seen the improvements in performance for the Log–APP algorithm compared to the Max–Log–APP algorithm.

Because the increase of the encoder memory order brings an improvement in the BER and FER performances, a study was made for this transmission system, where the encoder memory is increased from 2 to 3.

In Table 5 are given the FER and BER values obtained when the decoder uses the Log–APP algorithm, due to its superior performances. Also, there were given the obtained values from simulations only when the number of doubly iterations is k = 0 and k = 1, because for k = 4 there is a degradation in the performances obtained for memory 3, compared to those obtained when the encoder memory is 2, because of the errors introduced by the doubly iterations.

For k = 0, from Table 5 we see that for the case when the encoder memory is 3, there are two codes that lead to improved performances. The best BER values are obtained for code 13/15 (bold), and the best FER values are obtained for the code 15/13 (bold). The worst performances are given by the codes that have the non-primitive feedback polynomials equal to 17 in the octal form. One of these codes is 3/17 (italic).

We see again the slow performances of the non-primitive feedback polynomials, meaning that the p_2 polynomial from the form p_1/p_2 is 11 or 17. Nevertheless, we see that for the code 17/11 are obtained better BER and FER values, 9.68e–6 and 1.20e–3, respectively. A BER performance of order 10^{-4} and FER performance of order 10^{-2} , even if the feedback polynomial is non-primitive (11) is given by codes 3/11, 6/11, 13/11, 14/11 or 15/11.

For k = 1, we see that the best performances are given by the code 12/15. Also we can see that those codes that are composed by the primitive feedback polynomial 15 or 13 in octal form lead to good BER and FER performances. These codes are 12/13, 14/15, 14/13, 13/15, 15/13, etc. Also, the codes that have the primitive feedback polynomials 15 give better performances that those that have the primitive feedback polynomials 13. The worst performances are given by the code 3/17.

Generally, the codes that have the non-primitive feedback polynomials 11 or 17 lead to slow performances. As in the previous case, when k=0, we see that there are some codes with the non-primitive feedback polynomial 11 that give better performances, as the codes 6/11, 12/11, 13/11, 14/11 and 17/11.

For k = 0, the improvement of the performances brought by the increase of the memory order is clearly observed from the case when the decoder uses the same type of decoding algorithm, Log–APP. At SNR = –6 dB, for the codes that give the best performances for memory 3, namely 13/15 and 15/13, the BER and FER values are close to those obtained for the best code of memory 2, 5/7 at SNR = –5dB. At this value (SNR = –5 dB), for the code of memory 3, 15/13, simulations were performed for k = 0 and a number of blocks equal to 2,000,000 and were not found any errors. It results that the BER and FER performances obtained in the case of memory 3 are much better than in memory 2 case.

Table 5

Performances for Log–APP Algorithm for k = 0 and k = 1

	k = 0, SNR = –6 dB					k = 1, SNR = –6.5 dB				
p_1/p_2	BER	FER	*er*	*nber*	*nb*	BER	FER	*er*	*nber*	*nb*
1/13	4.55e-5	2.86e-3	1,651	50	17,445	6.16e-5	2.19e-3	2,916	50	22,780
1/15	4.01e-5	2.80e-3	1,488	50	17,828	6.63e-5	2.76e-3	2,498	50	18,113
1/17	1.17e-2	9.25e-1	1,315	50	54	8.23e-3	8.47e-1	1,009	50	59
2/11	1.39e-2	1.00e+0	1,451	50	50	1.08e-2	8.92e-1	1,267	50	56
2/13	4.60e-5	2.78e-3	1,718	50	17,962	1.16e-4	3.16e-3	3,832	50	15,804
2/15	5.02e-5	2.77e-3	1,879	50	18,017	1.06e-4	2.87e-3	3,842	50	17,363
3/11	7.10e-4	7.65e-2	964	50	653	2.71e-4	3.00e-2	938	50	1,666
3/13	1.06e-5	7.34e-4	1,506	50	68,059	2.24e-5	6.28e-4	3,711	50	79,601
3/15	7.66e-6	5.39e-4	1,474	50	92,605	1.75e-5	4.66e-4	3,903	50	107,102
3/17	*3.29e-2*	*1.00e+0*	*3,423*	*50*	*50*	*3.60e-2*	*1.00e+0*	*3,746*	*50*	*50*
4/11	1.36e-2	9.61e-1	1,472	50	52	1.19e-2	8.62e-1	1,442	50	58
4/13	5.16e-5	3.04e-3	1,762	50	16,426	1.31e-4	3.87e-3	3,524	50	12,901
4/15	3.88e-5	2.85e-3	1,411	50	17,489	8.04e-5	2.71e-3	3,079	50	18424
5/13	5.03e-6	3.47e-4	1,502	50	143,693	1.35e-5	3.23e-4	4,349	50	154,371
5/15	3.88e-6	2.78e-4	1,450	50	179,494	1.21e-5	3.29e-4	3,834	50	151,849
5/17	1.21e-2	9.61e-1	1,314	50	52	9.14e-3	7.04e-1	1,348	50	71
6/11	6.25e-4	6.46e-2	1,004	50	773	1.40e-4	2.02e-2	721	50	2,470
6/13	7.51e-6	5.16e-4	1,510	50	96,717	1.84e-5	5.94e-4	3,221	50	84,101
6/15	9.41e-6	5.95e-4	1,641	50	83,920	2.06e-5	4.60e-4	4,652	50	108,633
6/17	3.28e-2	1.00e+0	3,410	50	50	2.64e-2	1.00e+0	2,749	50	50
7/11	1.29e-2	9.61e-1	1,395	50	52	8.12e-3	8.19e-1	1,029	50	61
7/13	1.56e-5	1.34e-3	1,209	50	37,093	4.11e-5	1.22e-3	3,474	50	40,666
7/15	2.24e-5	1.38e-3	1,685	50	36,213	3.90e-5	1.22e-3	3,323	50	40,939
10/13	5.31e-5	2.64e-3	2,088	50	18,915	2.18e-6	8.66e-5	2,147	41	472,900
10/15	4.37e-5	2.46e-3	1,843	50	20,272	2.07e-6	8.10e-5	1,805	34	419,500
10/17	1.08e-2	9.43e-1	1,196	50	53	2.35e-3	3.93e-1	621	50	127
11/13	3.37e-6	1.48e-4	2,369	50	337,754	2.87e-6	6.17e-5	4,648	48	777,100
11/15	3.91e-6	1.28e-4	3,173	50	390,494	3.81e-6	5.85e-5	1,759	13	222,000
11/17	9.14e-3	1.00e+0	950	50	50	2.84e-3	8.62e-1	343	50	58
12/11	2.75e-3	2.29e-1	1,248	50	218	4.52e-5	5.19e-3	904	50	9,627
12/13	6.32e-6	3.69e-4	1,778	50	135,441	8.87e-7	2.76e-5	1,200	18	651,300
12/15	4.76e-6	3.35e-4	2,237	76	226,200	**6.47e-7**	**2.31e-5**	**464**	**8**	**345,100**
13/11	1.00e-4	3.44e-2	304	50	1,452	1.70e-6	5.27e-4	336	50	94,855
13/15	**1.40e-6**	**4.76e-5**	**1,225**	**20**	**419,900**	2.18e-6	4.96e-5	1,097	12	241,800
13/17	2.49e-4	9.52e-2	272	50	525	4.81e-6	2.17e-3	138	30	13,800
14/11	7.51e-4	6.89e-2	1,131	50	725	8.62e-6	1.14e-3	784	50	43,788
14/13	8.62e-6	7.66e-4	1,169	50	65,250	1.20e-6	2.81e-5	1,599	18	640,100
14/15	8.34e-6	4.70e-4	1,841	50	106,247	1.14e-6	3.16e-5	3,816	51	1,609,906
14/17	3.32e-2	1.00e+0	3,458	50	50	2.36e-2	1.00e+0	2,453	50	50
15/11	1.02e-4	4.09e-2	259	50	1,221	2.78e-6	5.38e-4	172	16	29,700
15/13	**1.61e-6**	**3.75e-5**	**1,611**	**18**	**480,000**	2.36e-6	4.77e-5	927	9	188,600
15/17	3.00e-4	7.27e-2	429	50	687	3.54e-6	1.42e-3	217	42	29,500
16/13	2.25e-5	1.36e-3	1,721	50	36,682	1.78e-6	1.98e-4	931	50	251,800
16/15	2.03e-5	1.40e-3	1,501	50	35,528	1.73e-6	6.55e-5	2,700	49	747,700
17/11	9.68e-6	1.20e-3	832	50	41,354	1.78e-6	5.66e-5	131	2	35,300
17/13	3.19e-6	8.53e-5	3,884	50	585,750	3.42e-6	6.01e-5	5,086	43	714,500

Comparing the obtained values for $k = 1$, for SNR = –6.5 dB, and Log–APP algorithm, for codes that give the best performance, *i.e*. 5/7 for memory 2 and 12/15 for memory 3, there can be seen the premise from which we started this analysis, meaning that the increased order of memory of the encoder improves the performances. Thus, we see that for the code 5/7, the obtained BER value is 1.21e–6 and for code 12/15 the BER value is 6.47e–7. Also, the FER value is 4.60e–5 for code 5/7 and 2.31e–5 for code 12/15.

5. Conclusions

The performances of different symmetric convolutional codes used in turbo decoding, in a limited-complexity doubly iterative decoder, are analysed. In the transmission system that presents a spatial interference canceling interface (an iterative MMSE receiver), the decoder uses a Ω'–QPP interleaver, and the simulations were performed for a Max–Log–APP and also a Log–APP turbo decoding algorithm. Also, the encoder memory is supposed to be 2 and 3. A doubly-iterative decoding process is used, scaling both the extrinsic information of the turbo decoder and the information at the input of the interference canceling block. Up to four iterations were used to cancel the spatial interferences. The simulation results show that the performances obtained in the case when the Log–APP algorithm is used, are much better that those obtained in the case of the Max–Log–APP algorithm. Also, the increase of the encoder memory to the order 3 leads to better FER and BER performances compared to those when the order of the encoder memory is 2. Another conclusion is that the codes that are built from feedback primitive polynomials lead to better performance than those with non-primitive feedback polynomials.

REFERENCES

Baltă H., Kovaci M., *The Performances of Convolutional Codes Used in Turbo Codes*. Bul. Şt. Univ. "Politehnica", Timişoara, S. Electron. şi Telecom., **49** (**63**), *2*, 38-43 (2004).

Biglieri E., Nordio A., Taricco G., *Doubly Iterative Decoding of Space-Time Turbo Codes with a Large Number of Antennas*. IEEE Trans. on Commun., **53**, *5*, 773-779 (2005).

Rotopănescu A.M., Trifina L., *Comparison between Ω'-QPP and Random Interleavers for Space-Time Turbo Codes Using Doubly Iterative Decoding Algorithm*. 10th Internat. Symp. on Signals, Circuits and Systems (ISSCS) 2011, Iaşi, România, June 30 – July 1, 2011, 411-414.

Rotopănescu A.M., Trifina L., *Doubly Iterative Decoding of Space-Time Turbo Codes Using Log-App Algorithm*. Acta Techn. Napoc. (Electronics and Telecommunications), **52**, *1*, 28-33 (2011).

Trifina L., Tărniceriu D., Rotopănescu A.M., *Influence of Extrinsec Information Scaling Coefficient on Doubly-Iterative Decoding Algorithm for Space-Time Turbo Codes with Large Number of Antennas*. Adv. in Electr. a. Comp. Engng., **11**, *1*, 85-90 (2011).

ASUPRA UNOR CODURI CONVOLUŢIONALE SIMETRICE DE MEMORIE DOI ŞI TREI FOLOSITE ÎN CODAREA TURBO SPAŢIO-TEMPORALĂ

(Rezumat)

Sunt analizate performanţele diferitelor coduri turbo convoluţionale simetrice pentru un sistem de transmisie cu diversitate de antene, care foloseşte un algoritm de decodare dublu iterativ. Folosind un interleaver Ω'–QPP şi doi algoritmi diferiţi de decodare, Max–Log–APP şi Log–APP, s-a constatat că performanţele obţinute în cazul în care este folosit cel de al doilea algoritm sunt mult mai bune decât cele obţinute în cazul în care se foloseşte algoritmul Max–Log–APP. De asemenea, creşterea memoriei codorului la ordinul 3 a condus la performanţe FER şi BER mai bune decât atunci când ordinul memoriei codorului este 2. Din simulări s-a observat că acele coduri care sunt construite după polinoame de reacţie primitive conduc la performanţe mai bune decât cele care sunt construite după polinoame de reacţie neprimitive.