Effect of Turbo Coding on OFDM Transmission to Improve BER

Dhiraj G. Agrawal¹, Roma K. Paliwal², Priti Subramaniam³

Abstract-A model of Turbo code based on OFDM (Orthogonal Frequency Division Multiplexing) is proposed in this paper. Orthogonal Frequency Division Multiplexing (OFDM) has become a popular modulation method in high speed wireless communications. By partitioning a wideband fading channel into flat narrowband channels, OFDM is able to mitigate the detrimental effects of multipath fading using a simple one-tap equalizer. There is a growing need to quickly transmit information wirelessly and accurately. OFDM is a suitable candidate for high data rate transmission with forward error correction (FEC) methods over wireless channels. In this project the system throughput of a working OFDM system has been enhanced by adding turbo coding. The use of turbo coding and power allocation in OFDM is useful to the desired performance at higher data rates. Simulation is to be done over additive white Gaussian noise (AWGN) and impulsive noise (which is produced in broadband transmission) channels. The wideband system has 48 data sub-channels; each is individually modulated according to channel state information acquired during the previous burst. This project is to increase the system throughput while maintaining system performance under a desired bit error rate (BER). To improve the performance of the uncoded OFDM signal by convolution coding[1].

Keywords: Bit error rate, Orthogonal frequency division multiplexing, Turbo codes,

I. INTRODUCTION

The telecommunications’ industry is in the midst of a veritable explosion in wireless technologies. Once exclusively military, satellite and cellular technologies are now commercially driven by ever more demanding consumers, who are ready for seamless communication from their home to their car, to their office, or even for outdoor activities. With this increased demand comes a growing need to transmit information wirelessly, quickly, and accurately. To address this need, communications engineers have combined technologies suitable for high rate transmission with forward error correction techniques. The latter are particularly important as wireless communications channels are far more hostile as opposed to wire alternatives, and the need for mobility proves especially challenging for reliable communications. Orthogonal Frequency Division Multiplexing (OFDM) is a Multi-Carrier Modulation technique in which a single high rate data-stream is divided into multiple low rate data-streams and is modulated using sub-carriers which are orthogonal to each other. Some of the main advantages of OFDM are its multi-path delay spread tolerance and efficient spectral usage by allowing overlapping in the frequency domain.

Also one other significant advantage is that the modulation and demodulation can be done using inverse fast fourier transmission (IFFT) and fast fourier transmission (FFT) operations, which are computationally efficient. In a single OFDM transmission all the subcarriers are synchronized to each other, restricting the transmission to digital modulation schemes. OFDM is symbol based, and can be thought of as a large number of low bit rate carriers transmitting in parallel. All these carriers transmitted using synchronized time and frequency, forming a single block of spectrum. This is to ensure that the orthogonal nature of the structure is maintained. Since these multiple carriers form a single OFDM transmission, they are commonly referred to as ‘subcarriers’, with the term of ‘carrier’ reserved for describing the RF carrier mixing the signal from base band. There are several ways of looking at what make the subcarriers in an OFDM signal orthogonal and why this prevents interference between them.

This project enhances the throughput of an existing OFDM system by implementing adaptive modulation and turbo coding. The new system guarantees to reach a target performance BER of 10⁻² over a slow time-varying fading channel. The system automatically switches from lower to higher modulation schemes on individual subcarriers, depending on the state of the quasi-stationary channel. In conjunction with the adaptive design, forward error correction is performed by using turbo codes. The combination of parallel concatenation and recursive decoding allows these codes to achieve near Shannon’s limit performance in the turbo cliff region [6]. All this is simulated in MATLAB programming.

The rest of the paper is organized as follows. In section II, we introduce OFDM transceiver model. In section III, briefly review the Turbo code design criteria. In section IV, The Simulation model is presented along with their implementation issues. The simulation results are presented in section V, and some conclusions and future work are drawn in section 6.

II. OFDM SYSTEM

Orthogonal frequency division multiplexing (OFDM) is nowadays widely used for achieving high data rates as well as combating multipath fading in wireless communications. In this multi-carrier modulation scheme data is transmitted by dividing a single wideband stream into several smaller or narrowband parallel bit streams.
Each narrowband stream is modulated onto an individual carrier. The narrowband channels are orthogonal vis-à-vis each other, and are transmitted simultaneously. In doing so, the symbol duration is increased proportionately, which reduces the effects of inter-symbol interference (ISI) induced by multipath Rayleigh-faded environments. The spectra of the subcarriers overlap each other, making OFDM more spectral efficient as opposed to conventional multicarrier communication schemes.

A. OFDM message

The OFDM message is generated in the complex baseband. Each symbol is modulated onto the corresponding subcarrier using variants of phase shift keying (PSK) or different forms of quadrature amplitude modulation (QAM). The data symbols are converted from serial to parallel before data transmission. The frequency spacing between adjacent subcarriers is $N \pi/2$, where $N$ is the number of subcarriers. This can be achieved by using the inverse discrete Fourier transform (IDFT), easily implemented as the inverse fast Fourier transform (IFFT) operation. As a result, the OFDM symbol generated for an $N$-subcarrier system translates into $N$ samples, with the $i$th sample being

$$X_i = \sum_{n=0}^{N-1} C_n \exp \left\{ \frac{2\pi i n}{N} \right\}, \quad 0 \leq i \leq N - 1$$

At the receiver, the OFDM message goes through the exact opposite operation in the discrete Fourier transform (DFT) to take the corrupted symbols from a time domain form into the frequency domain. In practice, the baseband OFDM receiver performs the fast Fourier transform (FFT) of the receive message to recover the information that was originally sent[2].

B. Interference

In a multipath environment, different versions of the transmitted symbol reach the receiver at different times. This is due to the fact that different propagation paths exist between transmitter and receiver. As a result, the time dispersion stretches a particular received symbol into the one following it. This symbol overlap is called intersymbol interference, or ISI. It also is a major factor in timing offset. One other form of interference is inter-carrier interference or ICI. In OFDM, successful demodulation depends on maintaining orthogonality between the carriers. We demodulate a specific subcarrier $N$ at its spectral peak, meaning that all the other carriers must have a corresponding zero spectra at the $N$th center frequency (frequency domain perspective). Frequency offsets lead to this criterion not being met. This condition can seriously hinder the performance of our OFDM system. Graph 3.1 below shows that when the decision is not taken at the correct center frequency (i.e. peak) of carrier considered, adjacent carriers factor in the decision making, thus reducing the performance of the system[2].

C. The Cyclic Prefix

![Figure 1 Basic OFDM System Architecture](image1)

![Figure 2 Effect of Frequency Offset (Maintaining Orthogonality)](image2)

![Figure 3. Cyclic Prefix](image3)
OFDM demodulation must be synchronized both in the time domain as well as in the frequency domain. Engineers have found a way to ensure that goal by adding a guard time in the form of a cyclic prefix (CP) to each OFDM symbol. The CP consists in duplicates of the end samples of the OFDM message relocated at the beginning of the OFDM symbol. This increase the length $T_{\text{sym}}$ of the transmit message without altering its frequency spectrum.

$$T_{\text{sym}} = \text{CP} + T_{\text{data}}N$$

where $T_{\text{data}}$ is the duration of one data symbol, and $N$ the number of carriers. The receiver is set to demodulate over a complete OFDM symbol period, which maintains orthogonality. As long as the CP, is longer than the channel delay spread, $\tau_{\text{max}}$, the system will not suffer from ISI. The CP is to be added after the FFT operation at the transmitter and removed prior to demodulation. The figure below whose the deterioration in performance when the CP is closely matched by the delay spread. The signal constellation is less tightly grouped, no doubt a sign of less than accurate decoding.

![Figure 4. 64QAM Signal Constellation Diagrams for a 64-Subcarrier OFDM System with Flat Rayleigh Fading. (A) The Cyclic Prefix is Long Enough to Cover The Delay Spread. (B) The Cyclic Prefix is Closer to Being Matched be the Delay Spread.](image)

III. TURBO CODES

The combination of turbo codes with the OFDM transmission is so called Turbo Coded OFDM (TC-OFDM) can yield significant improvements in terms of lower energy needed to transmit data, a very improvement issue is in personnel communication devices [1]-[2]-[20].

Turbo codes were first presented at the International Conference on Communications in 1993. Until then, it was widely believed that to achieve near Shannon’s bound performance, one would need to implement a decoder with infinite complexity or close. Parallel concatenated codes, as they are also known, can be implemented by using either block codes (PCBC) or convolutional codes (PCCC). PCCC resulted from the combination of three ideas that were known to all in the coding community:

- The transforming of commonly used non-systematic convolutional codes into systematic convolutional codes.
- The utilization of soft input soft output decoding. Instead of using hard decisions, the decoder uses the probabilities of the received data to generate soft output which also contain information about the degree of certainty of the output bits.
- This is achieved by using an interleaver. Encoders and decoders working on permuted versions of the same information.

An iterative decoding algorithm centered around the last two concept would refine its output with each pass, thus resembling the turbo engine used in airplanes. Hence, the name Turbo was used to refer to the process.

A. Encoders for Turbo Codes

The encoder for a turbo code is a parallel concatenated convolutional code. Figure 3.4 shows a block diagram of the encoder first presented by Berrou et al [10]. The binary input data sequence is represented by $d_k = (d_1, \ldots, d_0)$ the input sequence is passed into the input of a convolutional encoder [8], $\text{ENC}_1$, and a coded bit stream, $X_{k1}^p$, generated. The data sequence is then interleaved. That is, the bits are loaded into a matrix and read out in a way so as to spread the positions of the input bits. The bits are often read out in a pseudo random manner. The interleaved data sequence is passed to a second convolutional encoder $\text{ENC}_2$, and a second coded bit stream, $X_{k2}^p$, is generated. The code sequence that is passed to the modulator for transmission is a multiplexed (and possibly punctured) stream consisting of systematic code bits $X_{k1}^p$ and parity bits from both the first encoder $X_{k2}^p$ and the second encoder $X_{k2}^p$.

![Figure 5. Structure of a Turbo Encoder](image)

B. RSC Component Codes

$\text{ENC}_1$ and $\text{ENC}_2$ are Recursive Systematic Convolutional (RSC) codes that is, convolutional codes which use feedback (they are ‘recursive’) and in which the uncoded data bits appear in the transmitted code bit sequence (they are ‘systematic’). A simple RSC encoder is shown in Figure 3.5 along with a non-systematic (NSC) encoder, for comparison. The RSC encoder is rate 1/2, with constraint length $K=3$, and generator polynomial $G = \{g_1, g_2\} = \{7,5\}$ where $g_1$ is the feedback connectivity and $g_2$ is the output connectivity, in octal notation. An RSC component encoder has two output sequences. One is the data sequence; $X_{k}^p = \{x^p_{1}, \ldots, x^p_{8}\}$ the other is the parity sequence $X_{k}^o = \{x^o_{1}, \ldots, x^o_{1}\}$. 

96
To understand why RSC component codes are used in a turbo code encoder, rather than the conventional NSC codes, it is necessary to first discuss the structure of error control codes. The minimum number of errors an error control code can correct is determined by the minimum Hamming distance of the code – the minimum number of bit positions in which any two codewords differ. The linear nature of turbo codes (at least, those using BPSK/QPSK modulations) means that the minimum Hamming distance of the code can be determined by comparing each possible codeword with the all-zeroes codeword. This process simplifies analysis of the code somewhat, and the minimum Hamming distance is then equal to the minimum code weight (number of '1's) which occurs in any codeword. The minimum Hamming distance tends to determine the BER performance of the code at high SNR - the asymptotic performance. A high minimum Hamming distance results in a steep rate of fall of BER as SNR becomes large, whereas a low value results in a slow rate of fall. In the case of a turbo code, this rate of fall at high SNR is so slow; it is termed an error floor. At low SNR, however, codewords with code weights larger than the minimum value must be considered. It is then that the overall distance spectrum of the code becomes important.

![Diagram of Turbo Code Encoder](image)

**Figure 6. Example of (a) Non-Systematic Convolutional (NSC) (b) Recursive Systematic Convolutional (RSC) Encoders**

This is the relationship between the code weight and the number of codewords with that code weight. Now, RSC codes have an infinite impulse response. That is, if a data sequence consisting of a '1' followed by a series of '0's enters the RSC encoder, a code sequence will be generated containing both ones and zeroes for as long as the subsequent data bits remain zero. This property means that RSC encoders will tend to generate high weight code sequences for groups of data bits spread far apart in the input sequence.

An NSC code, however, will return to the all-zeroes state after K-1 input zeroes, where K is the constraint length of the encoder. The infinite impulse response property of RSC codes is complemented in turbo codes by the interleaver between component encoders. An interleaver is a device for permuting a sequence of bits (or symbols) at its input into an alternate sequence with a different ordering at the output.

Turbo codes tend to make use of pseudo-random interleavers, whose role is to ensure that most groups of data bits which are close together when entering one RSC encoder are spread far apart before entering the other RSC encoder. The result is a composite codeword which will often have a high code weight. The details of interleaving will be discussed in more detail in the next section. This does not, however, mean that turbo codes tend to exhibit high minimum Hamming distances, and therefore good asymptotic performance. In fact, the opposite is usually true. We shall see in the following chapter that the pseudo-random nature of most turbo code interleavers tends to result in a mapping such that a few combinations of input bit positions which cause low code weight sequences in one RSC component code are permuted into combinations of positions which generate low code weight sequences in the second RSC code. The result in such a case is a low composite code weight. Such pseudo-random mappings often lead to turbo codes having a low minimum code weight compared to, say, NSC-based convolutional codes, resulting in a marked error floor at high SNR. It is clear from this brief discussion that interleaver design is crucial in ensuring that a turbo code/interleaver combination has the lowest possible error floor. It was mentioned earlier that at low SNR, the distance spectrum of the code as a whole becomes significant in determining BER performance, and that the combination of RSC code and pseudorandom interleaving produces codewords with higher code weights most of the time. This results in there being fewer codewords with relatively low code weights than a comparable convolutional code. It shall be shown later in constructing theoretical bounds for turbo codes that it is the number of codewords at each weight, as well as the actual code weight, which determines the error probability of a code. The low multiplicity of low code weight sequences associated with turbo codes sometimes referred to as spectral thinning, leads to their good BER performance at low SNR. A full analysis of the turbo code characteristics described here is given in [8].

### C. Interleaving

It was mentioned in the previous section that an interleaver is a device for reordering a sequence of bits or symbols. A familiar role of interleavers in communications is that of the symbol interleaver which is used after error control coding and signal mapping to ensure that fading bursts affecting blocks of symbols transmitted over the channel are broken up at the receiver by a de-interleaver, prior to decoding. Most error control codes work much better when errors in the received sequence are spread far apart. Another role is that of the interleaver between component codes in a serially concatenated code scheme;
for example, between a Reed Solomon outer code and a convolutional inner code. The trellis decoding nature of most convolutional codes means that uncorrected errors at the output of the decoder will tend to occur in bursts. The interleaving between the two component codes then ensures that these bursts are adequately spread before entering the outer decoder. In both these examples, the interleaver is typically implemented as a block interleaver. This is a rectangular matrix such that bits or symbols are written in one row at a time, and then read out one column at a time. Thus bits or symbols which were adjacent on writing are spaced apart by the number of rows when reading. The de-interleaving process is simply the inverse of this; writing in column by column and reading out row by row, to achieve the original bit or symbol ordering. Block interleaving is simple to implement, and suitable where the objective is to spread bursts of errors evenly by as large a distance as possible. Block interleavers are not suitable as turbo code interleavers, because they tend to generate large numbers of codewords with a relatively low weight, and therefore with a relatively low hamming distance between them, due to the regularity of the spreading process. Berrou and Glavieux introduced pseudo-random interleaving into turbo codes to solve this problem. A pseudo-random interleaver is a random mapping between input and output positions, generated by means of some form of pseudo-random number generator. Figure 3.6 shows a simple illustration of pseudo-random interleaving. The original data sequence is represented by the sequence of white squares, and the interleaved data sequence is represented by the grey squares. Turbo code BER performance improves with interleaver length - the so-called interleaver gain - but the loading and unloading of the interleaver adds a considerable delay to the decoding process. This would make a 256x256 interleaver unsuitable for, say, and real time speech applications, which are delay sensitive.

**Figure 7. Pseudo-Random Interleaving in a Turbo Encoder**

**D. Puncturing**

Different code rates are achieved by puncturing the parity bit sequences $x_{k1}^p$ and $x_{k2}^p$. Puncturing the data bit sequence $x_k^d$ leads to a severe degradation in turbo code performance. Figure 3.4 illustrates the puncturing process. A number ‘1’ in the tables represents a code bit that is included in the transmitted code bit sequence, and a number ‘0’ represents a code bit that is excluded, or punctured. On the right of each table is the list of code bits which are included in the transmitted code sequence. In a), the code is unpunctured and is of code rate 1/3 whereas in b), alternate parity bits from each component encoder are punctured at each time interval $k$. The result is a rate 1/2 code. In c), the code is more heavily punctured, to form a rate codes 3/4.

**E. Termination**

In contrast to block codes, convolutional codes do not have a fixed block length. Convolutional coding is a continuous process, and could span an entire message, rather than a small group of bits. Turbo codes, however, do have a fixed block length, determined by the length of the interleaver. Tail bits are usually appended to each block of data bits entering one or other of the component encoders, to return it to the all-zeroes state at the end of the trellis. This process is called termination, and allows the MAP algorithm to make assumptions about the start and end trellis states. This yields better BER performance. Termination of both component encoders is more difficult, because the terminating sequence for the first encoder is interleaved and may well not, by itself, terminate the second encoder. Interleaver designs have been devised which interleave a terminating sequence in the first encoder into a terminating sequence in the second encoder. This tends to yield better BER performance than a single code terminating process and is another promising area for investigation.

**Figure 8. Puncture Patterns for Turbo Codes**
F. Turbo Decoding

A block diagram of a turbo decoder is shown in Figure 3.11. The input to the turbo decoder is a sequence of received code values \( R_k = \{ y_k^s, y_k^p \} \) from the demodulator. The turbo decoder consists of two component decoders – DEC1 to decode sequences from ENC1, and DEC2 to decode sequences from ENC2. Each of these decoders is a Maximum A Posteriori (MAP) decoder. DEC1 takes as its input the received sequence of systematic values \( y_k^s \) and the received sequence of parity values \( y_k^p_1 \) belonging to the first encoder ENC1. The output of DEC1 is a sequence of soft estimates \( \text{EXT1} \) of the transmitted data in that it does not contain any information which was given to DEC1 by DEC2. This information is interleaved, and then passed to the second decoder DEC2. The interleaver is identical to that in the encoder (Figure 3.1). DEC2 takes as its input the (interleaved) systematic received values \( y_k^s \) and the sequence of received parity values \( y_k^p_2 \) from the second encoder ENC2, along with the interleaved form of the extrinsic information EXT1, provided by the first decoder. DEC2 outputs a set of values, which, when de-interleaved using an inverse form of the interleaver, constitute soft estimates \( \text{EXT2} \) of the transmitted data sequence \( \hat{d}_k \). This extrinsic data, formed without the aid of parity bits from the first code, is feedback DEC1. This procedure is repeated in an iterative manner. The iterative decoding process adds greatly to the BER performance of turbo codes. However, after several iterations, the two decoders’ estimates of will tend to converge. At this point, DEC2 outputs a value \( \Lambda(\hat{d}_k) \); a log-likelihood representation of the estimate of \( \Lambda(d_k) \). This log likelihood value takes into account the probability of a transmitted ‘0’ or ‘1’ based on systematic information and parity information from both component codes. More negative values of \( \Lambda(d_k) \) represent a strong likelihood that the transmitted bit was a ‘0’ and more positive values represent a strong likelihood that a ‘1’ was transmitted. \( d_k \) is de-interleaved so that its sequence coincides with that of the systematic and first parity streams. Then a simple threshold operation is performed on the result, to produce hard decision estimates, \( \Lambda(d_k) \), for the transmitted bits.

The decoding estimates \( \text{EXT1} \) and \( \text{EXT2} \) do not necessarily converge to a correct bit decision. If a set of corrupted code bits form a pair of error sequences that neither of the decoders is able to correct, then \( \text{EXT1} \) and \( \text{EXT2} \) may either diverge, or converge to an incorrect soft value. In the next sections, we will look at the algorithms used in the turbo decoding process, within DEC1 and DEC2[20].

IV. THE SIMULATION MODEL

For simulation purposes, we based our work on the simulation tool provided online in [9]. It’s a complete OFDM WLAN physical layer simulation in MATLAB. The program simulates a 64 subcarrier OFDM system. The system supports up to 2 transmit and 2 receive antennas, a convolutional code generator with rates \( \frac{1}{2} \), \( \frac{2}{3} \), and \( \frac{3}{4} \). The code is punctured to IEEE specifications. As an option, one can choose to interleave the transmit bits for added protection. The system supports 4 modulation schemes, binary phase shift keying, quadrature phase shift keying, sixteen quadrature amplitude modulation, and sixty four quadrature amplitude modulation. Frequency jitter can also be added to a system that supports two channel models, namely additive white Gaussian noise, AWGN and flat Rayleigh fading. One can input the desired length of the delay spread. The cyclic prefix is 16 samples long. You can also request a specific average signal to noise ratio. Transmit power amplifier effects and phase noise distortion can be added to the transmit signal. The simulator also comes with a series of synchronization algorithms including packet detection, fine time synchronization, frequency synchronization, pilot phase tracking, channel estimation, all of that if you wish to simulate IEEE 802.11 standards. There is also a switch to add a receiver timing offset[1]-[2].

[99]
Here A = turbo encoder, B = QAM/QPSK modulation, C = serial to parallel converter, D = IFFT, E = parallel to serial converter, F = channel with noise, G = serial to parallel converter, H = FFT, I = parallel to serial converter, J = AM/QPSK demodulation and K = turbo decoder.

A. Simulation Parameters

During the simulations, in order compare the results, the same random messages were generated. For the radiant function is in MATLAB.

Table 3.4 Simulation Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Modulation</td>
<td>QPSK, 16-QAM 64-QAM</td>
</tr>
<tr>
<td>Turbo code rates</td>
<td>½</td>
</tr>
<tr>
<td>SISO Decoder</td>
<td>Log-MAP</td>
</tr>
<tr>
<td>Code Generator</td>
<td>{111, 101}</td>
</tr>
<tr>
<td>Interleaver Size</td>
<td>1 x 100</td>
</tr>
</tbody>
</table>

B. Algorithm of Simulation

We measured the performance of the turbo coded OFDM through MATLAB simulation. The simulation follows the procedure listed below:
1. Generate the information bits randomly.
2. Encode the information bits using a turbo encoder with the specified generator matrix.
3. Use QPSK or different QAM modulation to convert the binary bits, 0 and 1, into complex signals (before these modulation use zero padding)
4. Performed serial to parallel conversion.
5. Use IFFT to generate OFDM signals, zero padding is being done before IFFT.
6. Use parallel to serial convertor to transmit signal serially.
7. Introduce noise to simulate channel errors. We assume that the signals are transmitted over an AWGN channel. The noise is modeled as a Guassian random variable with zero mean and variance $\sigma^2$. The variance of the noise is obtained as

$$\sigma^2 = \frac{1}{2 * Eb/N0}$$

where randn has zero mean and 1 variance. Thus the

received signal at the decoder is $X' = \text{noisy (X)}$Where noisy (X) is the signal corrupted by noise.
8. At the receiver side, perform reverse operations to decode the received sequence.
9. Count the number of erroneous bits by comparing the decoded bit sequence with the original one.
10. Calculate the BER and plot it[1].

V. PERFORMANCE ANALYSIS

The majority of existing papers treating the TCOFDM assumes that the channel estimation using only by the pilot symbols is sufficient (or even that the channel is perfectly known). It is shown, however, that there is a large potential gain in using the iterative property of turbo decoders where soft bit estimates are used together with the known pilot symbols. The performance of such an iterative estimation scheme proves to be of particular interest when the channel is strongly frequency- and time- selective [1].

Similar to every other communications scheme, coding can be employed to improve the performance of overall system. Several coding schemes, such as block codes, convolutional codes and turbo codes have been investigated within OFDM systems. Moreover, the deep fades in the frequency response of the channel cause some groups of subcarriers to be less reliable than other groups and hence cause bit errors to occur in bursts rather than, independently. The burst errors can extensively degrade the performance of coding. To solve this problem several ways are considered. The easiest method is to use stronger codes, in fact an interleaving technique along with coding can guarantee the independence among errors by affecting randomly scattered errors. We use turbo code to improve the performance. For analysis of the OFDM system, first we examine uncoded situation and then we will analyze the effect of coding under turbo coded OFDM condition [1]-[2]-[6].

As mentioned before, bursty errors deteriorate the performance of the any communications system. The burst errors can happen either by impulsive noise or by deep frequency fades. Powerline channels suffer from both of these deficiencies. “Figure 11” shows the performance of uncoded OFDM system with AWGN and impulsive noise (which is modeled as marcov noise). In this “Figure 11” it is shown that, for the required BER $10^{-7}$ AWGN channel gives better performance as compared with marcov channel. AWGN gives a gain of approximately 22 db over marcov channel. We observe a little gain at lower SNR between 0 to <10dB, and more gain at higher SNR < 40dB[1].

To improve the performance of this system FEC code can be used. Convolutional code is a good example of FEC code. It is shown in “Figure 11”that Convolutional coding in OFDM can give performance improvement of some 5 db on AWGN channel over the uncoded OFDM system at required BER. Here the convolutional codes are based on the rate $\frac{1}{2}$, constraint length 3 and $(7, 5)$ generators matrix convolutional code.

Table 3.1 Comparison of SNR for Different Code Generators

<table>
<thead>
<tr>
<th>Code Generator</th>
<th>SNR for BER $10^{-2}$</th>
<th>SNR for BER $10^{-3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 5/7)</td>
<td>~ 7.2</td>
<td>~ 8.9 db</td>
</tr>
<tr>
<td>13</td>
<td>~ 6.8 db</td>
<td>~ 8.3 db</td>
</tr>
</tbody>
</table>

...
Further improvement in the performance can be obtained by applying turbo coding instead of convolutional code [13]. The turbo codes give better performance at low SNR. The BER performance of TCOFDM system is compared with the respective uncoded system under the fading AWGN channel. Simulating the turbo codes with polynomial generators, (1, 15/13)8 and (1, 5/7)8 which are iteratively decoded by Log-MAP for a number of decoding iterations. The simulated results are shown in Figure 12. From the results, we observe that both turbo codes (1, 15/13)8 and (1, 5/7)8 give considerably good BER performance[1]. Comparing (1, 15/13)8 codes with (1, 5/7)8, we observe a little gain at higher SNR between 8 to <10 dB.

Table 3.2 Comparison of Turbo Coded and Convolutional Coded OFDM over Uncoded

<table>
<thead>
<tr>
<th>Type of Coded OFDM</th>
<th>Gain at $10^2$ over Uncoded OFDM</th>
<th>Gain at $10^3$ over Uncoded OFDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional coded OFDM</td>
<td>4.8 db</td>
<td>5.2 db</td>
</tr>
<tr>
<td>16QAM TCOFDM</td>
<td>6.5 db</td>
<td>7.5 db</td>
</tr>
<tr>
<td>QPSK TCOFDM</td>
<td>11.5 db</td>
<td>13 db</td>
</tr>
</tbody>
</table>

Broadband communications for indoor powerline networks with impulsive noise using OFDM is considered. Here simulation is done on two type of impulsive noise model.

The first is marcov and second is asynchronous impulsive noise is modeled by the fact that the time domain.

VI. CONCLUSION

To conclude, this major project gives the detail knowledge of a current key issue in the field of communications named Orthogonal Frequency Division Multiplexing (OFDM). We focused our attention on turbo codes and their implementation. We described the encoder architecture. In our case, the code is the result of the parallel concatenation of two identical RSCs. The code can be punctured in order to fulfill bit rate requirements. The decoder succeeded in its duty thanks to the decoding algorithms that it is built around. We focused mainly on the study of the MAP.
We discovered that the power of the scheme came from the two individual decoders performing the MAP on interleaved versions of the input. Each decoder used information produced by the other as a priori information and outputted a posteriori information. We elaborated on the performance theory of the codes. Then we tied concepts of OFDM and turbo coding with a target-based, modulation scheme. First I developed an OFDM system model then try to improve the performance by applying forward error correcting codes to our uncoded system. From the study of the system, it can be concluded that we are able to improve the performance of uncoded OFDM by convolutional coding scheme. Further improvement on the performance has been achieved by applying turbo coding to uncoded OFDM system. Turbo codes with low order decoding iterations have been evaluated. The SNR performance for BER 10^{-2} and 10^{-3}, that are suitable for speed and data applications, are analyzed. As a result, the TCOFDM system with least number of decoding iterations, 3 to 5 iterations are shown to be sufficient to provide good BER performance.

REFERENCES


Author’s Details

Dhiraj G Agrawal¹, Asst. Professor, S. S. G. B. C. O. E. & Technology, Bhusawal, dg_agrawal@rediffmail.com
Roma K. Paliwal², M.E. Student, S. S. G. B. C. O. E. & Technology, Bhusawal, rpaliwal199@gmail.com
Priti Subramanium³, Asst. Professor, S. S. G. B. C. O. E. & Technology, Bhusawal, pritikanna559@gmail.com

1 © 2007 IEEE

Department of Electronics

2 M.E. Student, S. S. G. B. C. O. E. & Technology, Bhusawal, paliwal199@gmail.com

3 Asst. Professor, S. S. G. B. C. O. E. & Technology, Bhusawal, pritikanna559@gmail.com