RESULT AND DISCUSSION
OBSERVATIONS AND RESULT

4.1 Interleaved Parameter Design for Turbo Codes

This is widely accepted by the international research community that the interleaver parameter in coding plays a very important role in the construction of good Turbo codes. There are valid indications as to what makes a good interleaver design; unfortunately, however, there is no comprehensive explanation or computable quantification of the effects of interleaver choice on the resultant Turbo code. In this lesion, we draw together known results and attempt to use these to determine which parameters make a good interleaver design.

4.2 Interleaving in Coding Systems

4.2.1 Classical Use of Interleavers parameter

This type of interleavers has been used in communication systems for a long time. The main use is to randomize the location of errors. It enables the use of random error correcting codes on channels coding scheme with burst error patterns of the coding. The burst channels would include fading channels in wireless transmission system.

Another use of interleaving in coding is in concatenated coding, where the output of the outer-stage decoder in system exhibits burst error patterns. Therefore, the most important parameter of the interleaver in this coding scheme is its ability to spread error in the form of bursts in random manner such that it appear as isolated errors forms to the inner stage of decoder in system. Therefore, the optimum interleaver may achieve this type of performance with the minimum memory size in coding. It is scene that the required parameters of the interleaver depend on the inner code, outer codes and decoders used in system [22, 23].

4.2.2 Interleavers in Turbo Codes

The Turbo coding is also introduced a further dimension to what is required from the interleaver. This involves the main effects of the iterative algorithm in coding scheme and the passing of intrinsic information between two successive decoder stages of turbo code. In this type of context, the interleaver has been explained by reducing the correlation between the parity bits corresponding to the original and interleaved data frames. Due to this reasons it
will be explained later in this chapter, this type of terminology is not quite accurate terminology in coding scheme.

The original paper of Turbo codes introducing by Berrou and Glavieux [1993] already showed an exceptional understanding of some of the most important parameters which makes the good interleaver. In particular case

I. Increasing the block size or size of the interleaver results in improved performance in coding scheme [31,32]

II. The interleaver is used to randomize the input sequence in order to avoid particular low-weight patterns mapping onto themselves. It reduces the effective free distance of the resulting Turbo Code in system.

The interleavers used with Turbo Codes are block interleavers. This type of interleavers can be described by a $T \times T$ matrix. Where every row and every column contains a single 1 and $T - 1$ zeros. If a 1 occurs in the $i^{th}$ row and the $j^{th}$ column, then the interleaver moves the $i^{th}$ input symbol to the $j^{th}$ output position. Recently, however, another kind of interleaver has re-emerged and the convolutional interleaver may be classically consist of a set of $T$ shift registers of increasing length, with the input sequence being multiplexed into $T$ subsequences, thus introducing a different delay for each subsequence. The output sequence is obtained by de-multiplexing the outputs of the shift registers. This type of interleavers has been applied to stream-oriented Turbo codes by Hall & Wilson [1998].

### 4.3 Performance of Turbo Codes in coding

Before discussing of this type of coding what makes a good interleaver type design, we will briefly analyze the performance of Turbo codes in my system. It is based on their distance spectrum. This analysis assumes a ML decoder in system. It is published by Perez et al. and Benedetto and Montorsi. The iterative decoder used in Turbo codes is not a ML decoder but its performance is very close, so that the analysis is a valid explanation of Turbo code performance in coding scheme.

#### 4.3.1 Recursive and Non-Recursive Codes

The part played by the RSC codes as components for Turbo codes is very critical.

Although NRC codes perform better than RSC codes when it is used on their own the particular weight distribution of RSC codes results in a significant improvement within the Turbo coding system. For example, the 2-state systematic NRC code with generator $(1; 3)$ and RSC code with generator $(1; 2, 3)$. When used with a MAP decoder with block size $= 100$
(including a 1-bit tail), the BER performance of the NRC code is superior to RSC code, as shown in Fig. 4.1. The difference between the two codes is immediately apparent when we consider the weight distribution of the terminated convolutional codes. The weight distributions for NRC and RSC codes up to a codeword weight of 20 are shown in Figs. 4.2 and 4.3 respectively. On the graphs darker regions denote a higher multiplicity, where multiplicity is the number of codeword’s with the specified input and output weight. In each weight of the input sequence of coding, the weight of codeword’s generated by the RSC coding span a broad range with a rather uniform multiplicity scheme. On the other hand, for input sequences with low weight, the NRC code generates a small set of codeword weights of low value. In a Turbo code, the interleaved sequence has the same weight at the input sequence in permuting interleaver. The order of the bits is changed and weight of the parity bits for the interleaved sequence may be change from the input sequence. In NRC codes, the low-weight inputs sequence is associated with low-weight parity sequences in coding scheme. It means that input and interleaved sequences will have a low weight in coding. On the other hand, if RSC codes are used in coding, there is good chance that inputs associated with low-weight parity may be mapped to an interleaved sequence associated with high-weight parity bits in coding scheme of turbo code.

![Figure 4.1: Performance of 2-state RSC and NRC codes](image-url)
### Table 4.1 Performance of 2-state RSC and NRC codes

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER PERFORMANCE</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER PERFORMANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NRC</td>
<td></td>
<td></td>
<td>RSC</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>6×10^{-2}</td>
<td>10</td>
<td>4.5</td>
<td>1.5×10^{-3}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9×10^{-2}</td>
<td></td>
<td></td>
<td>3×10^{-3}</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>5×10^{-2}</td>
<td>11</td>
<td>5.0</td>
<td>0.5×10^{-3}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7×10^{-2}</td>
<td></td>
<td></td>
<td>1.5×10^{-3}</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>3×10^{-2}</td>
<td>12</td>
<td>5.5</td>
<td>7×10^{-4}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5×10^{-2}</td>
<td></td>
<td></td>
<td>0.5×10^{-3}</td>
</tr>
<tr>
<td>4</td>
<td>1.5</td>
<td>2×10^{-2}</td>
<td>13</td>
<td>6.0</td>
<td>3×10^{-4}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.5×10^{-2}</td>
<td></td>
<td></td>
<td>7×10^{-4}</td>
</tr>
<tr>
<td>5</td>
<td>2.0</td>
<td>2×10^{-2}</td>
<td>14</td>
<td>6.5</td>
<td>1×10^{-4}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4×10^{-2}</td>
<td></td>
<td></td>
<td>1.5×10^{-4}</td>
</tr>
<tr>
<td>6</td>
<td>2.5</td>
<td>0.8×10^{-2}</td>
<td>15</td>
<td>7.0</td>
<td>4×10^{-5}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.8×10^{-2}</td>
<td></td>
<td></td>
<td>1×10^{-4}</td>
</tr>
<tr>
<td>7</td>
<td>3.0</td>
<td>1×10^{-2}</td>
<td>16</td>
<td>7.5</td>
<td>1.5×10^{-5}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2×10^{-2}</td>
<td></td>
<td></td>
<td>3×10^{-5}</td>
</tr>
<tr>
<td>8</td>
<td>3.5</td>
<td>7×10^{-3}</td>
<td>17</td>
<td>8.0</td>
<td>1×10^{-5}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1×10^{-2}</td>
<td></td>
<td></td>
<td>0.8×10^{-5}</td>
</tr>
<tr>
<td>9</td>
<td>4.0</td>
<td>4.0×10^{-3}</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.0×10^{-3}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 4.3.2 Performance Bound

The BER of a linear block code can be upper bounded by the union bound in equation given below

\[
P_b(w, d) \cdot Q \leq 4.1\end{equation}

Here \(A(w, d)\) is the number of codeword’s with input weight \(w\) and total weight \(d\). The code’s block size is represented by the number of information bits \(\tau\) and the code rate \(R\). We assume that the tail length \(\nu \leq \tau\). we can use \(\tau\) as being the length of the whole source sequence in coding system with tail bits.

changing the order of summation in coding

\[
P_b(w,d) \cdot Q \leq 4.2\end{equation}

\[
\sum_{d} Q \leq 4.3\end{equation}
The term $A_d$ is the total information weight of all codeword’s of weight $d$ and it is divided by number of information bits per codeword. It is defined as:

$$A_d = (w, d).$$ \hspace{1cm} 4.4

Where $N_d$ to be the number of codeword’s of total weight $d$. $W_d$ is a average information weight in coding.
Thus,

\[ N_d \cdot W_d = \quad 4.5 \]

\[ A_d = w_d. \quad 4.6 \]

The ratio \( N_d/\tau \) is known as effective multiplicity of codeword’s of weight \( d \).

### 4.3.3 Free Distance and Performance at High SNR

It is seen in Eqn. (4.6) that the BER contribution of codeword’s of weight \( d \) depends on three point

![Figure 4.3: Weight distribution pattern of terminated RSC code](image)
1. Their average information weight $W_d$.
2. Their effective multiplicity $N_d$.
3. The complementary error function $Q$

The first two terms depend on the code structure and final term depends on the channel SNR. In order to visualize its effect in coding, we plot the value of the $Q$ function of typical range of channel SNR and for codeword weights up to $d = 20$. The code rate is $1/3$. It is seen that as the SNR increases, the contribution due to larger codeword weights is reduced and BER performance of the code is dominated factor by the free distance at high SNR. Thus for Turbo codes the asymptotic performance approaches to the

$$P_b \approx w_{\text{free}} \cdot Q$$

The term $N_{\text{free}}$ is the multiplicity of free distance of codeword’s and $w_{\text{free}}$ is the average weight of information sequences. The Turbo code with a random interleaver, a few low-
weight input sequences with low-weight parity bit may be mapped to interleaved sequences and it is also associated with low-weight parity bit. This type of codeword’s lower the free distance of the Turbo code, causing the error floor in coding scheme.

4.3.4 The Spectral Thinning and Performance at Low, Medium SNR

For uniform interleaver in turbo code, the several low weight input sequences with low-weight parity will be mapped to other interleaved sequences which are associated with high-weight parity bit. The average parity weight is high even for low-weight input. It is compared with the weight distribution of its component RSC code, the number of low-weight input sequences in coding associated with high-weight parity bit is increased while those associated with a low-weight parity bit remain few. This phenomenon is known as spectral thinning because the low-weight portion of the distance spectrum in coding retains a low multiplicity. The free distance of a Turbo code may be small compared with a Maximum Free Distance (MFD) convolutional code of larger memory in coding system. The multiplicity of low-weight codeword’s is lower. This means that at low-medium SNR, when the effect of higher-weight codeword’s cannot be neglected, the performance of a Turbo code is better. This phenomenon was first explained by Perez et al.

4.3.5 Interleaver Input-Output Distance Spectrum parameter

For visualize the spread of an interleaver and how it is achieved in coding, we plot the Input-Output Distance Spectrum (IODS) with graph. This type of graph is a two dimensional histogram graph, with the axes being distance between all possible bit pairs of the input and output of the interleaver. The darker the region the more instances with the given parameters. For comparison purpose, we plot the IODS (limiting ourselves to a distance of 100) for a Square and Berrou-Glavieux interleaver of equal size in Fig. 4.5 and 4.6 respectively. These interleavers are used in next Chapter.

Note how the Square interleaver in coding concentrates bit-pairs in very small regions, while the Berrou-Glavieux interleaver through the pseudo-random perturbations spreads out the regions. Due to this reason, the peak frequency for the Square interleaver is almost four times that of the Berrou-Glavieux interleaver in system.
4.4 Interleaver Requirements

While restricting ourselves to block interleaver structures in coding, which can describe any interleaver used in a block-oriented Turbo code in system, we discuss below the main requirements of interleaver design in coding system. The simultaneous optimization of all requirements in coding is a problem of significant complexity; unfortunately, even their separate optimization is non-trivial because of inter-dependant restrictions in coding system.
4.4.1 Block Size parameter

The length of the interleaver design should be as large as possible for improved performance in turbo coding. It can be seen from Eqn. (4.7) that as block size increases (with free distance and multiplicity are unchanged) the effective multiplicity of the free distance term is decreased. This lowers the error floor asymptote without changing its slope, in proportion with the increase in block size. This effect has been suitably presented and quantified by the JPL team. Note, however, that increasing Output Distance the interleaver size also increases the decoder delay by a corresponding amount in turbo code.

4.4.2 Mapping Randomization parameter

The importance of randomization in the interleaver design is evident and accepted. One of the reasons given for randomization is the breaking of low-weight sequences of code to increase
the free distance spectrum. It improves the code’s asymptotic performance. The effect of randomization as a method of reducing correlation between the parity sequences in turbo code, and its effects on the iterative nature of the Turbo decoder is still unclear. This reason is close to the classical use of interleavers, allowing the use of random error-correcting codes in situations where there is a marked temporal correlation in noise patterns. The requirement for randomization as opposed to spreading is still unclear in this sense of coding [20, 21].

4.4.3 Spreading parameter

In order to spread error bursts between successive decoders, the interleaver must ensure that bit pairs which are close in the original sequence will be further apart in the interleaved sequence. The limit is chosen because in a convolutional code, an error in the received sequence will affect the decoder’s performance for bits within this distance of the error. This can be visualized by observing the confidence at the output of a MAP decoder when the input has a single hard error. Fig 4.7. shows the confidence at the output of a MAP decoder for every bit in a frame consisting of 102 data bits and 2 tail bits. The code used is a RSC code with polynomials (1; 5=7). The confidence metric is computed as the ratio \( p_0/p_1 \), where \( p_0 \) and \( p_1 \) are the \textit{a posteriori} probabilities of the correct and incorrect bit, respectively, at a given position within the frame. Notice how a single hard error at position 52 affects approximately ten bits on either side. As one would expect, closer bits are affected most; the effect practically vanishes beyond the distance of bits. Other points worth noting are that tail and lead bits have a higher confidence (because in those cases the decoder knows the start/final state with certainty), and that the confidence graph is symmetric about the point containing the error.
A quantification of the effect of interleaver spread on the Turbo code’s performance is still missing. Also, the effect of multiplicity of points in the IODS close to the spread boundary is not known.
4.4.4 Punctured Code Interleavers

Some work has been done on the effect of puncturing in turbo coding. In particular type, the JPL groups have been computed a list of the best rate 1/2 constituent codes with help of odd-even puncturing scheme. It is based on the minimum weights of parity sequences in low-weight information sequences of code. Another contribution in this field, this time due to Barbulescu, is the restriction imposed on the interleaver design to make it odd-even pattern.

4.4.5 Trellis Type Termination

The original Turbo code ignored the effect of terminating the information sequences in system. This effect is considered insignificant for large block sizes in coding system. It may have a detrimental effect on small codes. A sufficient condition on the interleaver to ensure that both the original and the interleaved sequence are terminated is given by Blackert et al. 1995. A further improvement in coding scheme is given by Barbulescu 1996; by further restricting the interleaver, both the original and interleaved sequence have the same tail bit and the same parity sequence. This effectively removes the effect of puncturing in the tail section of coding.

4.5 Different Parameter Analysis

The different types of analysis in coding can be subdivided into three categories. The standard interleaves types include those used with Turbo codes in the literature. The Reference interleavers have been designed with the primary intention of gaining some insight into the problem, third, the Optimized interleavers includes our efforts at interleaver design in coding and others already mentioned in our literature.

4.5.1 Standard Interleavers parameter

4.5.1.1 Rectangular Interleaver parameter

The classical block interleaver has well-known spread properties in turbo coding scheme. Ramsey proves that we can construct an optimal interleaver with a spread of (R, C). It creates a rectangular interleaver with R rows and C columns, where information is written row-wise and read column-wise. It must be satisfied following conditions given below:

I. \( R + 1 \) and \( C \) are relatively prime
II. \( R + 1 < C \). Note that a Square interleaver cannot obey this condition.
We can formalize the mapping function for such a rectangular interleaver by the equation:

\[(t) = [t \mod R]. C + (t \div R)\]  \hspace{1cm} (4.8)

**Theorem 4.5.1.1** A rectangular interleaver is odd-even if \(R\) and \(C\) are both odd.

**Proof.** Consider a rectangular interleaver as defined by previous Eqn. Now, let

\[i = t \mod R\]  \hspace{1cm} (4.9)

\[j = t \div R\]

Here, we can rewrite the interleaver mapping as

\[(t) = i.C + j\]  \hspace{1cm} (4.10)

\[t = j \cdot R + i\]  \hspace{1cm} (4.11)

an odd-even type interleaver, \((t)\) must be even if \(t\) is even, and odd if \(t\) is odd. Thus, \(C\) must be even for all values of \(t\) in equation. Now, for the rectangular interleaver considered,

\[=\]  \hspace{1cm} (4.12)

\[=\]  \hspace{1cm} (4.13)

It is ensure that \((C - 1)\) and \((R - 1)\) must be both even type. Hence, \(R\) and \(C\) must be odd type.

The rectangular interleave creates a problems in implementation when used in a Turbo code. The certain input patterns with weight \(w = 4\) is written in the rectangular interleaver. The four 1’s are at the corners of a square. The sides of which have lengths equal to a multiple of \(p\). it is length of the impulse response of the RSC component code. An example is shown below. For such cases, the parity sequences for both the interleaved and non-interleaved stream will have
a low weight in coding system. It is impairing the performance of the resulting Turbo code. This phenomenon is documented in the literature of Berrou & Glavieux, [1996].

Write

\[
\begin{matrix}
0 & 0 & . & . & . & 0 & 0 & 0 \\
0 & . & . & . & . & . & . & 0 \\
. & 1 & 0 & 0 & 1 & . & . & . \\
. & 0 & 0 & 0 & 0 & . & . & . \\
. & 0 & 0 & 0 & 0 & . & . & . \\
. & 1 & 0 & 0 & 1 & . & . & . \\
. & . & . & . & . & . & . & . \\
0 & . & . & . & . & . & 0 \\
0 & 0 & 0 & . & . & . & 0 & 0 \\
\end{matrix}
\]

Example of weight 4 sequence, not broken by a rectangular interleaver

4.5.1.2 Berrou-Glavieux Interleaver parameter

The interleaver used in the first Turbo code by Berrou and Glavieux to appear in the literature is a rectangular interleaver modified to avoid the known flaws. For an interleaver memory (\( m \) being a power of two), the all information is written by row-wise and read pseudo-randomly as described by equation

\[
\begin{align*}
i_r &= (i + j) \quad \text{ (mod) } \quad 4.14 \\
j_r &= [P(j) - j + 1] - 1 \quad \text{ (mod) } \quad 4.16
\end{align*}
\]

Where \( i, j \) are respectively the row and column for writing purpose and \( i_r, j_r \) are the row and column for reading purpose. The interleaver mapping is given by equation
P(ξ) is a number, relatively prime with , which is a function of the line address ξ.

For all cases where is a power of two, the function P (ξ) as given below in Table 4.3 may be used. This function is used in the original Turbo code by Berrou et al. [1993], it is first documented in a subsequent paper [1996]. It is noted that reading is performed diagonally in order to avoid possible effects of a relation between and the period of puncturing. Also, a multiplying factor (/2 + 1) is used to prevent two neighboring data. It is written on two consecutive lines from remaining neighbors in reading. Together with the pseudo-randomness introduced by P(ξ), this avoids the input weight w = 4 with a problem involved with the rectangular interleaver design.

Table 4.3: Pseudo Random Function for Interleaver

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>37</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
</tr>
</tbody>
</table>

4.5.1.3 Helical Interleaver parameter

It is another modification of the rectangular interleaver, it was originally proposed by Barbulescu [1996]. The helical interleaver is made by R rows and C columns. The data is written row-wise and read diagonally in system with starting from the bottom-left entry.

\[ t = i . C + j \]
\[ i_r = C \cdot R - 1 \cdot t \pmod{R} \]  
\[ j_r = t \pmod{C} \]  

It is clear that the interleaver is odd-even if \( C \) is even and the interleaver is Correctly-terminating if \( C \) is a multiple of \( v + 1 \) for RSC constituent codes with a ‘full’ feedback polynomial. A further restriction on the interleaver construction is that \( R \) and \( C \) have to be relatively prime no in interleaver design.

### 4.5.2 Reference Interleavers

#### 4.5.2.1 Uniform Interleaver parameter

It is defined by average of all possible interleavers with random choice with a uniform distribution. It is very useful interleaver because a number of bounds actually assume this kind of interleaver. The simulation of Turbo code with a uniform interleaver enables a fair comparison of such bounds in coding scheme. It gives a desirable performance that should be achievable without too much effort in coding system. The formal definition of such an interleaver implies that the performance of the resulting Turbo code may be computed by averaging the results of Turbo codes with all possible interleavers design. But this solution is impractical and there is \( \tau! \) Different interleavers in a block size of \( \tau \), we can obtained a statistically valid result by simulating a large number of blocks with several different interleavers in turbo code. In our implementation, we simulate a Turbo code with a different random interleaver for each frame. It means that:

I. We assume that the sought result actually exists in system and mean of the samples converges as the number of samples taken is increased in coding system.

II. The distribution of the random interleavers is flat and all interleavers are equally probable for any frame being simulated in coding scheme.

The simulation time is increased for a uniform interleaver. It is compared with other Turbo codes with the same parameters but a fixed interleaver. We have not taken as encountered cases where assumption taken in the system is not valid. The simulations agree with bounds which assume a uniform interleaver.

#### 4.5.2.2 Flat Interleaver parameter
The flat interleaver simply replicates the incoming sequence in coding. This is the worst choice for an interleaver, but may be used to gain insight into the interleaver design problem. The mapping function for such an interleaver is trivial and represented by

\[ (t) = t \]

Usually the flat interleaver is always odd even and it is correctly-terminating.

**4.5.2.3 Barrel-Shifting Interleaver parameter**

The Barrel-Shifting interleaver is used to separate the interleaved bits from the original bits without disturbing their order in coding scheme. The permutation function of this interleaver is simply to transpose all input bits by a fixed number of positions into output sequence with wrapping at the frame boundary. It is defined by equation given below

\[ (t) = t + \theta \text{ (mod } N) \] \hspace{1cm} 4.21

Here \( \theta \) is represented by block size and \( \xi \) is represented by shift distance. This interleaver can be made odd even by choosing \( \xi \) to be even. Also, for a correctly-terminating interleaver, \( \xi \) needs to be a multiple of \( p \) in this case, the period of the RSC component code.

**4.5.2.4 One-Time Pad Interleaver parameter**

Rather than shuffling the information sequence in time, the One-Time Pad (OTP) interleaver adds a known random sequence with the one-time pad to the information sequence. In this type, any correlation between the parity sequences associated with the straight and interleaved frames is broken and the interleaver delay is zero. The OTP interleaver does not shuffle the bits along the time axis. This type of interleaver must necessarily be odd-even type.
Procedure s-random
begin
  Generate an empty interleaver array of size
  for i = 0 to
    repeat
      Generate the random integer in (0,)
    until distance between generated integer and s previously generated integer is greater than s
  end for
end

Semi Random Interleaver Design algorithm

4.5.3 Optimized Interleavers parameter
4.5.3.1 Semi-random Interleaver parameter

The all Turbo code interleaver design techniques in the literature are based on the Semi-random interleaver generation algorithm. It is proposed by the JPL team Divsalar & Pollara. This interleaver design is basically randomly chosen interleaver with a restriction on its spreading. The algorithm for choosing an interleaver with spread semi is given below. The searching time of algorithm increases with semi S and it is not fully guaranteed to finish successfully in system. Divsalar and Pollara have been observed that choosing S usually produces a result in a reasonable time period. The main problems with these techniques are that it is not guaranteed to produce the required interleaver and that it only aims at achieving a spreading S. it is seen in IODS space. It means that a square of side S starting at the origin must be kept empty and the distribution of points outside this square is not considered. Usually, with a large interleaver size that is randomly chosen, the distribution outside the spreading boundary is approximately flat. It is seen in Fig. 4.8, it shows the IODS for an S-random interleaver of size τ = 1024. Compared with interleavers having higher regularity, such as the Berrou-Glavieux interleaver of Fig. 4.6, the flattening of the IODS reduces the peak frequency in the interleaver. The peak frequency of the Berrou-Glavieux interleaver is about eight times that of the semi random interleaver type.
4.5.3.2 Simulated Annealing Interleaver parameter

To optimize the various requirements, the one possibility is to define an energy function based on those requirements and use the simulated annealing algorithm. It is not guaranteed to converge to an optimal solution, in our experience it can be used with success in various optimization processes with large dimensionality. This type of an interleaver would have the advantage of appearing similar to a random one. In investigations of coding, the number of error functions in coding scheme has been tested in simulation. The main design criterion in our case is a modification of the 'high spreading factor' criterion used by the JPL team in their semi random interleaver type. Here we have aimed that to obtained high spreading factor at least $5\nu$ in coding, where $\nu$ is defined as component code’s memory order, we also obtained the worst case spreading to give low multiplicity in coding scheme. The limit of $5\nu$ is obtained after analyzing the confidence for output of a MAP decoder when the input has a single hard error in coding scheme. It is not very difficult to construct the interleaver in order to make it
correctly terminating interleaver type. Since this design technique works directly on the interleaver’s Input-Output Distance Spectrum (IODS) rather than restricting only particular input-output distance conditions, it should result in improved performance, because we are not exclusively considering weight-2 input sequences confidence at the output of a MAP decoder when the input had a single hard error. Procedure for simulated annealing as follows.

Begin
Initialize state $X = X_0$
Initiaise temperature $T = T_0$
Repeat
  Repeat
    Choose $X'$, a perturbation of $X$
    Let $E = \text{Energy}(X') - \text{Energy}(X)$
    If $E \leq 0$ or random (0,1)
      Change state $X = X'$
  End if
Until several state change or too many iteration
Update temperature according to annealing schedule
Until final temperature reached or configuration is stable
End

Simulated Annealing – Basic algorithm

It is not difficult to constrain the interleaver in order to make it correctly-terminating type. This design technique actually works on the interleaver’s Input-Output Distance Spectrum (IODS) rather than restricting only particular input-output distance conditions of interleaver, it should result in improved performance, because we are not exclusively considering weight-2 input sequences in design.

4.5.3.2.1 Interleaver Design with Simulated annealing parameter
The Simulated Annealing (SA) algorithm for global minimization is a well-proven and accepted technique in interleaver design and it has been used successfully in various applications for minimization of a combinatorial function. The method used in this technique is a close analogy with thermodynamics, particularly with the way that slowly cooled crystals...
achieve a state of minimum energy function. The minimization algorithm implemented in our case is the Metropolis Algorithm in interleaver design, where a downhill move is always performed and uphill move is accepted with a probability \( E_{\text{new}} - E_{\text{old}} / 2 \) where \( T \) is the temperature, \( E_{\text{old}} \) is the energy before the move and \( E_{\text{new}} \) is the energy after the move being considered in system. In our case, we design interleavers according to a predefined set of requirements by define an energy function based on those criteria. Temperature is a virtual value which is decreased exponentially. The basic algorithm is outlined in Fig. The parameters that can be varied with

1. the energy function of system
2. the perturbation scheme
3. the initial code assignment \( X_0 \)
4. the initial and final temperatures
5. the annealing schedule
6. the number of iterations allowed at every temperature
7. the number of state changes allowed per temperature (if less than the maximum number of iterations)

It is preferred to have a perturbation scheme in coding and energy function is chosen that the variation of energy at every perturbation be as smooth as possible. The perturbation scheme must be selected so as not to exclude any part of the search space. We have attempted to meet these conditions in our case by restricting perturbations to a swap of two random interleaver systems. The initial assignment \( X_0 \) should not make any difference in the result if the other parameters are selected correctly. This is because at the initial temperature, the system’s energy will rise considerably as most perturbations will be accepted. A random interleaver should be as good as any other interleaver. This has been verified experimentally in my research. To allow the system to migrate towards a high-energy level in the initial phase of system, the starting temperature should be selected such that \( T_0 - \Delta E_{\text{avg}} \). This ensures that uphill changes have a high probability of occurrence at initial temperatures level. It is note that the choice of a good perturbation scheme is necessary such that \( \Delta E \) has a sufficiently restricted variance in system. The algorithm stops when the state configuration becomes stable. This can be implemented by stopping the algorithm after five reductions in temperature occur without any new state being accepted. In this case, this is enough, but a minimum temperature condition is usually applied in system. This minimum temperature should be selected such
that it is sufficiently lower than the temperatures at which the configuration becomes stable in the normal case of system. Since the probability of acceptance of an uphill move decreases exponentially with temperature variation, the annealing schedule of system is generally a geometric decrease of temperature $T_{i+1} = \alpha T_i$, where $\alpha$ lies in the range 0:90 to 0:99. Values of .99 closer to 1:0 make the temperature decrease more slowly and it results in better annealing scheme, with a penalty to be paid as an increased simulation time.

The maximum number of iterations to be performed at every temperature level should be large enough to allow the configuration to reach an energy state typical of that temperature level. In practice, the number of iterations should be such that a significant proportion of the possible perturbations are tried. The only adverse effect of choosing too high a value is a proportional increase in computation time. It may be noted that this value is particularly very important at low temperatures level, where very few state changes are in fact accepted in system. At high temperatures level, most proposed changes will be accepted in system, so the necessary number of iterations is lower at high temperatures level. This is achieved by adding another limit to the number of iterations .it changes the maximum number of accepted state, which should be lower than the maximum number of iterations in coding, typically by an order of magnitude. But, at high temperatures, the number of iterations performed is limited by the number of accepted state changes, while at lower temperatures; it is limited by the maximum number of iterations in coding. To improve the performance of the resulting Turbo code, we
need to minimize the number of points close to the origin in the interweaver’s IODS. With this in mind, we choose an energy function which sums over all points in the IODS, with increased weight on points close to the origin. This attempts to ‘push‘bit-pair points away from the origin; it increases the spread of the interleaver. One particular energy function that can be used is:

\[ E = \quad 4.22 \]

Where \( i, j \in [0, \tau - 1], i > j, \tau \) is the interleavers size, \( \nu \) is the encoder memory, and Output Distance \( \lambda \) is the interleaving function. Note that the denominator is the radial distance from the origin of the point described by the bit pair \( i; j \).

Figure 4.9: IODS for Simulated Annealing Interleaver
4.5.3.2.2 Comparison of IODS with Semi-Random Interleavers

In contrast with the JPL technique used in turbo coding, our algorithm does not guarantee a particular spreading. On the other way, it pushes points away from the origin point even beyond the spreading boundary condition. The IODS for an interleaver designed by using Simulated Annealing is shown in Fig. 4.10. For the comparison purpose, I will see also the IODS for an semi random interleaver, designed using the JPL algorithm. It is noted that the IODS peak frequency of the Simulated Annealing interleaver is comparable with that of the semi random interleaver.

4.5.3.2.3 Correctly-Terminating Interleavers

Interleaver to be correctly-terminating, the mapping function must satisfy the equation

\[ t = (t) \mod p \]

here \( p \) is the encoder’s periodicity. For designing correctly-terminating interleavers by using simulated annealing scheme, it is sufficient condition to:

I. Create an initial interleaver which satisfies Eqn.

II. Modify the perturbation function to choose two random positions \( t_1 \) and \( t_2 \) which satisfy the equation:

\[ t = (t) \mod p \]

It is noted that these conditions are tied to the component codes used in coding’ parameters, and it restrict the search space to the set of correctly-terminating interleavers. It makes it harder to increase the interleaver spreading. The IODS for two small interleavers designed by using simulated annealing with and without the correct-termination restriction are given in Figs. 4.10 and 4.11 respectively. The interleavers are used for codes H and F respectively. It is listed in Table. It can be seen from the graphs how the restriction of termination degrades the interleaver spread in coding scheme.

4.5.3.2.4 Odd-Even Interleavers Type

For an odd-even interleaver, the mapping function must satisfy the equation
where \( s \) is the number of sets of Turbo code. It is the number of parallel component codes. This type of interleaver is used to perform better in the presence of puncturing scheme. For both odd-even and correctly-terminating interleaver, it must satisfy the equation

\[
t = (t) \quad (\text{mod } s)
\]

Figure 4.10: IODS for a Small Simulated Annealing Interleaver (Self-Terminating)
Figure 4.11: IODS for a Small Simulated Annealing Interleaver (Non-Terminating)

\[ t = (t) \mod(s,p) \]  

Where \( (a; b) \) gives the lowest common multiple of \( a \) and \( b \). Since our analysis is restricted in Turbo codes without puncturing the data, this modification to the simulated annealing algorithm has not been implemented in system.
4.6 Interleaver Design for Large Frames

4.6.1 Basic

In this lesion we have consider the interleaver design problem for large block sizes in coding scheme, where the effect of trellis termination is less marked. It is done by comparing the performance of different interleavers with a similar block size used in turbo code. The novel interleaver schemes are also used in coding. In which we gather some further insight into the problem. Finally, the performance of an optimized interleaver design based on simulated annealing scheme is considered in turbo code [75].

4.6.2 Performance Reference

Figure 4.11: Turbo code Bit Error Rate simulation with large block size and Uniform interleaver type

We restrict ourselves for unpunctured code rate 1/3 in symmetric Turbo codes with encoder memory ν= 2 and generator (1; 5/7). In order to avoid the effects of trellis termination in coding, we have chosen a relatively large block size with τ = 1024 bit. The effect of interleaver choice for Turbo codes with component code and block size are indicative of what can be expected with other good component codes and larger block sizes. The design problem of choosing the best component codes has been tackled by Divsalar & Pollara. With help of
performance basis, we have considered a uniform interleaver by using a different type of random interleaver for every block simulated in coding system. To check the validity of this statement in different type of interleaver design that trellis termination does not have a significant effect at the selected block size.

We have simulated the uniform interleaver with and without termination in turbo coding scheme. For termination of uniform interleaver, we have used the scheme that is proposed by the JPL team in coding system. The Bit Error Rate (BER) and Frame Error Rate (FER) performance results for these two Turbo codes (in Table 4.4) are shown respectively in Figs. 4.11 and 4.12; all simulations were performed by using 10 iterations in coding, with a target tolerance of ±10% at a confidence level of 95%. These tolerance limits are shown in the graphs. As expected, trellis termination does not significantly affect the performance of Turbo codes with encoder memory \( v = 2 \) at this block size of turbo code.

![Figure 4.12: Turbo code with FER simulation of large block size and Uniform interleaver](image-url)
Table 4.4: A large block size and Uniform interleaver

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.5</td>
<td>$1.0 \times 10^{-1}$</td>
<td>$1.0 \times 10^{-9}$</td>
<td>12</td>
<td>0.6</td>
<td>$3.0 \times 10^{-3}$</td>
<td>$1.8 \times 10^{-4}$</td>
</tr>
<tr>
<td>2</td>
<td>-0.4</td>
<td>$1.0 \times 10^{-1}$</td>
<td>$9.8 \times 10^{-1}$</td>
<td>13</td>
<td>0.7</td>
<td>$2.0 \times 10^{-3}$</td>
<td>$1.5 \times 10^{-4}$</td>
</tr>
<tr>
<td>3</td>
<td>-0.3</td>
<td>$9.0 \times 10^{-2}$</td>
<td>$9.5 \times 10^{-1}$</td>
<td>14</td>
<td>0.8</td>
<td>$9.0 \times 10^{-4}$</td>
<td>$7.0 \times 10^{-2}$</td>
</tr>
<tr>
<td>4</td>
<td>-0.2</td>
<td>$8.0 \times 10^{-2}$</td>
<td>$9.0 \times 10^{-1}$</td>
<td>15</td>
<td>0.9</td>
<td>$4.0 \times 10^{-4}$</td>
<td>$5.0 \times 10^{-2}$</td>
</tr>
<tr>
<td>5</td>
<td>-0.1</td>
<td>$7.0 \times 10^{-2}$</td>
<td>$8.5 \times 10^{-1}$</td>
<td>16</td>
<td>1.0</td>
<td>$2.0 \times 10^{-4}$</td>
<td>$4.0 \times 10^{-2}$</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>$5.0 \times 10^{-2}$</td>
<td>$8.0 \times 10^{-1}$</td>
<td>17</td>
<td>1.1</td>
<td>$1.0 \times 10^{-4}$</td>
<td>$3.0 \times 10^{-2}$</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>$4.0 \times 10^{-2}$</td>
<td>$7.0 \times 10^{-1}$</td>
<td>18</td>
<td>1.2</td>
<td>$7.0 \times 10^{-5}$</td>
<td>$2.5 \times 10^{-2}$</td>
</tr>
<tr>
<td>8</td>
<td>0.2</td>
<td>$3.0 \times 10^{-2}$</td>
<td>$6.0 \times 10^{-1}$</td>
<td>19</td>
<td>1.3</td>
<td>$6.0 \times 10^{-5}$</td>
<td>$2.0 \times 10^{-2}$</td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td>$2.0 \times 10^{-2}$</td>
<td>$4.5 \times 10^{-1}$</td>
<td>20</td>
<td>1.4</td>
<td>$5.0 \times 10^{-5}$</td>
<td>$1.8 \times 10^{-2}$</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
<td>$1.0 \times 10^{-2}$</td>
<td>$3.5 \times 10^{-1}$</td>
<td>21</td>
<td>1.5</td>
<td>$4.0 \times 10^{-5}$</td>
<td>$1.5 \times 10^{-2}$</td>
</tr>
<tr>
<td>11</td>
<td>0.5</td>
<td>$7.0 \times 10^{-3}$</td>
<td>$2.5 \times 10^{-1}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.6.3 Regular Interleavers Type

It is clear that interleavers with a high regularity give poor performance in Turbo coding. It is clear in Figs. 4.13 and 4.14 where we simulate the Turbo codes with Square, Rectangular, and Helical interleavers (codes a, b, and c in Table 4.11). For comparison point of view, the performance of a uniform interleaver (code e) is also shown in the graphs. Note that the Rectangular and Helical interleavers satisfy all the restrictions detailed by Ramsey, Barbulescu & Pietrobon, respectively.

**Rectangular Type:** The interleaver with $R = 21$ rows and $C = 49$ columns, $R + 1$ and $C$ are relatively prime and $R + 1 < C$.

I. The interleaver spread is greater than $5v$.

II. Additionally, $R$ and $C$ are both odd, so that the interleaver is odd-even.

This allows the same interleaver to be fairly compared with other such interleavers in a study of punctured Turbo codes in coding scheme.

Table For Rectangular Interleaver is given below

Table 4.5: large block size and Regular interleavers type

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.5</td>
<td>$0.2 \times 10^{-1}$</td>
<td>$1.0 \times 10^{0}$</td>
<td>12</td>
<td>0.6</td>
<td>$9.0 \times 10^{-3}$</td>
<td>$1.8 \times 10^{-1}$</td>
</tr>
<tr>
<td>2</td>
<td>-0.4</td>
<td>$1.0 \times 10^{-1}$</td>
<td>$9.8 \times 10^{-1}$</td>
<td>13</td>
<td>0.7</td>
<td>$6.0 \times 10^{-3}$</td>
<td>$1.5 \times 10^{-1}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-0.3</td>
<td>9.0×10^{-2}</td>
<td>9.5×10^{-1}</td>
<td>14</td>
<td>0.8</td>
<td>3.0×10^{-3}</td>
<td>7.0×10^{-2}</td>
</tr>
<tr>
<td>4</td>
<td>-0.2</td>
<td>8.0×10^{-2}</td>
<td>9.0×10^{-1}</td>
<td>15</td>
<td>0.9</td>
<td>1.9×10^{-3}</td>
<td>5.0×10^{-2}</td>
</tr>
<tr>
<td>5</td>
<td>-0.1</td>
<td>7.0×10^{-2}</td>
<td>8.5×10^{-1}</td>
<td>16</td>
<td>1.0</td>
<td>0.9×10^{-3}</td>
<td>1.2×10^{-1}</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>5.0×10^{-2}</td>
<td>8.0×10^{-1}</td>
<td>17</td>
<td>1.1</td>
<td>8.×10^{-4}</td>
<td>0.8×10^{-1}</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>4.0×10^{-2}</td>
<td>7.0×10^{-1}</td>
<td>18</td>
<td>1.2</td>
<td>7.0×10^{-4}</td>
<td>0.6×10^{-1}</td>
</tr>
<tr>
<td>8</td>
<td>0.2</td>
<td>3.0×10^{-2}</td>
<td>6.0×10^{-1}</td>
<td>19</td>
<td>1.3</td>
<td>6.0×10^{-4}</td>
<td>0.4×10^{-1}</td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td>2.0×10^{-2}</td>
<td>4.5×10^{-1}</td>
<td>20</td>
<td>1.4</td>
<td>5.0×10^{-4}</td>
<td>0.3×10^{-1}</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
<td>0.9×10^{-2}</td>
<td>3.0×10^{-1}</td>
<td>21</td>
<td>1.5</td>
<td>4.0×10^{-4}</td>
<td>9.0×10^{-2}</td>
</tr>
<tr>
<td>11</td>
<td>0.5</td>
<td>0.5×10^{-2}</td>
<td>1.8×10^{-1}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**

- Uniform (Unterminated)
- Square (32x32)
- Rectangular (21x49)
- Helical (20x38)
Figure 4.13: Turbo code BER simulation with large block size and Regular interleavers type

Helical Type: The interleaver with \( R = 29 \) rows and \( C = 36 \) columns and \( R \) and \( C \) are relatively prime.

I. We know that \( C \) is a multiple of \( \nu + 1 \) and the RSC code’s feedback polynomial is full. It makes the interleaver simile. This allows the same tail bit to be used with both the interleaved and non-interleaved sequences in coding. In our implementation, we use the Tail Not Interleaved scheme proposed by Barbulescu, which is shown to give better results in coding scheme.

II. If \( C \) is even, so that the interleaver is odd-even. This allows the same interleaver to be fairly compared with other interleavers in a study of punctured Turbo codes. It is clear that the BER performance of turbo code improvement of the helical interleaver over the Square and Rectangular interleavers is minimal, and it can probably be attributed mostly to the termination. Its FER performance is significantly better, though still far from the uniform interleaver in turbo code.
Table 4.6: For Helical Interleaver

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.5</td>
<td>0.2×10^2</td>
<td>1.0×10^0</td>
<td>12</td>
<td>0.6</td>
<td>9.0×10^3</td>
<td>3.2×10^1</td>
</tr>
<tr>
<td>2</td>
<td>-0.4</td>
<td>0.9×10^1</td>
<td>9.7×10^1</td>
<td>13</td>
<td>0.7</td>
<td>6.0×10^3</td>
<td>2.5×10^1</td>
</tr>
<tr>
<td>3</td>
<td>-0.3</td>
<td>8.0×10^2</td>
<td>9.3×10^1</td>
<td>14</td>
<td>0.8</td>
<td>3.0×10^3</td>
<td>1.8×10^1</td>
</tr>
<tr>
<td>4</td>
<td>-0.2</td>
<td>7.0×10^2</td>
<td>8.9×10^1</td>
<td>15</td>
<td>0.9</td>
<td>1.9×10^3</td>
<td>0.9×10^1</td>
</tr>
<tr>
<td>5</td>
<td>-0.1</td>
<td>6.0×10^2</td>
<td>8.4×10^1</td>
<td>16</td>
<td>1.0</td>
<td>0.1×10^3</td>
<td>0.8×10^1</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>4.5×10^2</td>
<td>8.2×10^1</td>
<td>17</td>
<td>1.1</td>
<td>7.0×10^4</td>
<td>0.3×10^1</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>3.5×10^2</td>
<td>8.1×10^1</td>
<td>18</td>
<td>1.2</td>
<td>6.0×10^4</td>
<td>7.8×10^2</td>
</tr>
<tr>
<td>8</td>
<td>0.2</td>
<td>2.5×10^2</td>
<td>8.0×10^1</td>
<td>19</td>
<td>1.3</td>
<td>5.0×10^4</td>
<td>5.4×10^2</td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td>1.8×10^2</td>
<td>7.5×10^1</td>
<td>20</td>
<td>1.4</td>
<td>4.0×10^4</td>
<td>4.0×10^2</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
<td>0.8×10^2</td>
<td>5.3×10^1</td>
<td>21</td>
<td>1.5</td>
<td>3.0×10^4</td>
<td>3.8×10^2</td>
</tr>
<tr>
<td>11</td>
<td>0.5</td>
<td>0.4×10^2</td>
<td>4.8×10^1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.6.4 Randomised Interleavers Type

The interleaver basically used by Berrou & Glavieux, 1996.it is essentially a Square interleaver with pseudo random perturbations. Its performance is significantly better than a regular Square interleaver with the same dimensions. A comparison between the Berrou-Glavieux interleaver (code d) and the Square interleaver (code a) is shown in Figs. 4.14 and 4.15. For comparison, the performance of a uniform interleaver (code E) is also shown in the graphs.

4.6.5 Analysis of Bad Interleavers

4.6.5.1 Barrel-Shifting Interleave

Before creating the optimized interleaver, we identify two parameters that do not improve performance in turbo code. The first parameter is increasing the distance between a bit’s position in the input and its position in the interleaved stream (i.e. |λ(t) − t|) of coding scheme.

Table 4.7: Glavieux (32 ×32) with large block size, Randomised interleavers type

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.5</td>
<td>0.2×10^1</td>
<td>1.0×10^0</td>
<td>12</td>
<td>0.6</td>
<td>2.0×10^3</td>
<td>0.5×10^1</td>
</tr>
<tr>
<td>2</td>
<td>-0.4</td>
<td>0.9×10^1</td>
<td>9.8×10^1</td>
<td>13</td>
<td>0.7</td>
<td>0.5×10^3</td>
<td>9.0×10^2</td>
</tr>
<tr>
<td>3</td>
<td>-0.3</td>
<td>9.5×10^2</td>
<td>9.3×10^1</td>
<td>14</td>
<td>0.8</td>
<td>7.0×10^4</td>
<td>4.0×10^2</td>
</tr>
<tr>
<td></td>
<td>BER</td>
<td>Eb/No (dB)</td>
<td>BER</td>
<td>Eb/No (dB)</td>
<td>BER</td>
<td>Eb/No (dB)</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>------------</td>
<td>------</td>
<td>------------</td>
<td>------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.2</td>
<td>9.0 × 10^{-2}</td>
<td>8.0 × 10^{-1}</td>
<td>15</td>
<td>0.9</td>
<td>2.5 × 10^{-4}</td>
<td>2.0 × 10^{-2}</td>
</tr>
<tr>
<td>5</td>
<td>-0.1</td>
<td>8.0 × 10^{-2}</td>
<td>7.4 × 10^{-1}</td>
<td>16</td>
<td>1.0</td>
<td>1.1 × 10^{-4}</td>
<td>1.0 × 10^{-2}</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>7.5 × 10^{-2}</td>
<td>7.2 × 10^{-1}</td>
<td>17</td>
<td>1.1</td>
<td>8.0 × 10^{-5}</td>
<td>0.5 × 10^{-2}</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>4.5 × 10^{-2}</td>
<td>6.1 × 10^{-1}</td>
<td>18</td>
<td>1.2</td>
<td>6.0 × 10^{-5}</td>
<td>9.0 × 10^{-3}</td>
</tr>
<tr>
<td>8</td>
<td>0.2</td>
<td>2.0 × 10^{-2}</td>
<td>5.0 × 10^{-1}</td>
<td>19</td>
<td>1.3</td>
<td>2.5 × 10^{-5}</td>
<td>5.5 × 10^{-3}</td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td>1.0 × 10^{-2}</td>
<td>4.0 × 10^{-1}</td>
<td>20</td>
<td>1.4</td>
<td>1.2 × 10^{-5}</td>
<td>3.5 × 10^{-3}</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
<td>0.3 × 10^{-2}</td>
<td>2.0 × 10^{-1}</td>
<td>21</td>
<td>1.5</td>
<td>0.5 × 10^{-5}</td>
<td>2.0 × 10^{-3}</td>
</tr>
<tr>
<td>11</td>
<td>0.5</td>
<td>6.0 × 10^{-3}</td>
<td>1.2 × 10^{-1}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.14: Turbo code BER simulation with large block size and Randomised interleavers type
Figure 4.15: Turbo code FER simulation with large block size and Randomized interleaver type

Table 4.8: large block size and Barrel-shift interleaver type

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER Uniform</th>
<th>BER Flat</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER Uniform</th>
<th>BER Flat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.5</td>
<td>0.2×10^-1</td>
<td>9.9×10^-1</td>
<td>12</td>
<td>0.6</td>
<td>3.0×10^-3</td>
<td>8.5×10^-2</td>
</tr>
<tr>
<td>2</td>
<td>-0.4</td>
<td>0.9×10^-1</td>
<td>9.8×10^-1</td>
<td>13</td>
<td>0.7</td>
<td>0.8×10^-3</td>
<td>8.0×10^-2</td>
</tr>
<tr>
<td>3</td>
<td>-0.3</td>
<td>9.5×10^-2</td>
<td>9.3×10^-1</td>
<td>14</td>
<td>0.8</td>
<td>7.0×10^-4</td>
<td>7.5×10^-2</td>
</tr>
<tr>
<td>4</td>
<td>-0.2</td>
<td>9.0×10^-2</td>
<td>8.0×10^-1</td>
<td>15</td>
<td>0.9</td>
<td>3.0×10^-4</td>
<td>7.0×10^-2</td>
</tr>
<tr>
<td>5</td>
<td>-0.1</td>
<td>8.0×10^-2</td>
<td>7.4×10^-1</td>
<td>16</td>
<td>1.0</td>
<td>0.3×10^-4</td>
<td>6.5×10^-2</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>7.5×10^-2</td>
<td>7.2×10^-1</td>
<td>17</td>
<td>1.1</td>
<td>8.0×10^-5</td>
<td>6.0×10^-2</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>4.5×10^-2</td>
<td>6.1×10^-1</td>
<td>18</td>
<td>1.2</td>
<td>9.0×10^-5</td>
<td>5.5×10^-2</td>
</tr>
<tr>
<td>8</td>
<td>0.2</td>
<td>2.0×10^-2</td>
<td>5.0×10^-1</td>
<td>19</td>
<td>1.3</td>
<td>5.0×10^-5</td>
<td>5.0×10^-2</td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td>1.0×10^-2</td>
<td>4.0×10^-1</td>
<td>20</td>
<td>1.4</td>
<td>3.8×10^-5</td>
<td>4.5×10^-2</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
<td>0.3×10^-2</td>
<td>9.5×10^-2</td>
<td>21</td>
<td>1.5</td>
<td>3.0×10^-5</td>
<td>4.0×10^-2</td>
</tr>
<tr>
<td>11</td>
<td>0.5</td>
<td>6.0×10^-3</td>
<td>9.0×10^-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The barrel-shifting interleaver specifies this distance (\( \zeta \)) while keeping everything else the same as in the input stream. We compare the Bit Error Rate performance of this interleaver for various values of \( \zeta \) with help of flat interleaver and a uniform interleaver in Fig. 4.16. It is note that how the performance of turbo code is very poor for such an interleaver type and increasing \( \zeta \) has negligible effect in coding system. It is also note that how the performance of this interleaver is practically identical to that of a flat interleaver. We have considered as a special case of barrel shift interleaver.(\( \zeta = 0 \)). The FER performance of interleaver is not shown because the flat and barrel-shifting interleavers have a FER of 100% within the SNR range considered in coding scheme.

4.6.5.2 One-Time Pad Interleaver Type

The second interleaver is considered as the one time pad interleaver (OTP) interleaver, where the input stream is not permuted in time domain, but rather has a random sequence added to it.
Table 4.9: large block size and One Time Pad interleaver type

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>BER</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>BER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Uniform</td>
<td>Flat</td>
<td></td>
<td></td>
<td>Uniform</td>
<td>Flat</td>
</tr>
<tr>
<td>1</td>
<td>-0.5</td>
<td>0.2×10⁻¹</td>
<td>9.8×10⁻¹</td>
<td>12</td>
<td>0.6</td>
<td>2.9×10⁻³</td>
<td>8.7×10⁻²</td>
</tr>
<tr>
<td>2</td>
<td>-0.4</td>
<td>0.9×10⁻¹</td>
<td>9.2×10⁻¹</td>
<td>13</td>
<td>0.7</td>
<td>0.7×10⁻³</td>
<td>8.2×10⁻²</td>
</tr>
<tr>
<td>3</td>
<td>-0.3</td>
<td>9.5×10⁻²</td>
<td>8.5×10⁻¹</td>
<td>14</td>
<td>0.8</td>
<td>6.0×10⁻⁴</td>
<td>7.7×10⁻²</td>
</tr>
<tr>
<td>4</td>
<td>-0.2</td>
<td>9.0×10⁻²</td>
<td>7.3×10⁻¹</td>
<td>15</td>
<td>0.9</td>
<td>2.0×10⁻⁴</td>
<td>7.2×10⁻²</td>
</tr>
<tr>
<td>5</td>
<td>-0.1</td>
<td>8.0×10⁻²</td>
<td>6.7×10⁻¹</td>
<td>16</td>
<td>1.0</td>
<td>0.2×10⁻⁴</td>
<td>6.8×10⁻²</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>7.5×10⁻²</td>
<td>6.5×10⁻¹</td>
<td>17</td>
<td>1.1</td>
<td>7.0×10⁻⁵</td>
<td>6.4×10⁻²</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>4.5×10⁻²</td>
<td>5.3×10⁻¹</td>
<td>18</td>
<td>1.2</td>
<td>8.0×10⁻⁵</td>
<td>6.5×10⁻²</td>
</tr>
<tr>
<td>8</td>
<td>0.2</td>
<td>2.0×10⁻²</td>
<td>4.3×10⁻¹</td>
<td>19</td>
<td>1.3</td>
<td>4.9×10⁻⁵</td>
<td>5.5×10⁻²</td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td>1.0×10⁻²</td>
<td>5.2×10⁻¹</td>
<td>20</td>
<td>1.4</td>
<td>3.7×10⁻⁵</td>
<td>4.9×10⁻²</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
<td>0.3×10⁻²</td>
<td>8.6×10⁻²</td>
<td>21</td>
<td>1.5</td>
<td>2.9×10⁻⁵</td>
<td>4.4×10⁻²</td>
</tr>
<tr>
<td>11</td>
<td>0.5</td>
<td>6.0×10⁻³</td>
<td>8.1×10⁻²</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.17: Turbo code BER simulation with large block size and One-time pad interleaver type

The performance of a random One Time Pad interleaver is compared to a flat type and a uniform type) interleaver in Fig. The average performance of all OTP interleavers, computed by using a renewable One Time Pad interleaver is also included in the graph. As for the barrel-
shift interleaver, the performance of the One Time Pad interleaver is very poor. It is being practically identical to the flat interleaver type. This means that it is not the lack of correlation between input and interleaved sequences in coding that gives a Turbo code its good performance in coding scheme.

4.6.6 Optimized Interleaver Design in Turbo code Parameter

The performance of Simulated Annealing interleaver of Turbo code is shown in Figs. 8.9 and 8.10. For comparison purpose, codes of the uniform interleaver and the Berrou-Glavieux interleaver are also shown in figure. It is seen from the graphs, optimized interleaver improves performance of the Turbo code in the form of both BER and FER. We can achieve a BER of $10^{-5}$ at $E_b/N_0 = 1.35$ dB in this type of interleaver.

Table 4.10: large block size and Optimized interleavers type

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.5</td>
<td>0.1×10^{-1}</td>
<td>1.0×10^{-6}</td>
<td>12</td>
<td>0.6</td>
<td>2.0×10^{-3}</td>
<td>1.0×10^{-1}</td>
</tr>
<tr>
<td>2</td>
<td>-0.4</td>
<td>0.8×10^{-1}</td>
<td>0.8×10^{-6}</td>
<td>13</td>
<td>0.7</td>
<td>0.5×10^{-3}</td>
<td>6.0×10^{-2}</td>
</tr>
<tr>
<td>3</td>
<td>-0.3</td>
<td>9.0×10^{-2}</td>
<td>9.3×10^{-1}</td>
<td>14</td>
<td>0.8</td>
<td>5.0×10^{-4}</td>
<td>4.0×10^{-2}</td>
</tr>
<tr>
<td>4</td>
<td>-0.2</td>
<td>8.5×10^{-2}</td>
<td>8.3×10^{-1}</td>
<td>15</td>
<td>0.9</td>
<td>1.0×10^{-4}</td>
<td>1.8×10^{-2}</td>
</tr>
<tr>
<td>5</td>
<td>-0.1</td>
<td>7.5×10^{-2}</td>
<td>7.3×10^{-1}</td>
<td>16</td>
<td>1.0</td>
<td>1.0×10^{-5}</td>
<td>1.0×10^{-2}</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>7.0×10^{-2}</td>
<td>7.1×10^{-1}</td>
<td>17</td>
<td>1.1</td>
<td>3.0×10^{-5}</td>
<td>8.0×10^{-3}</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>3.5×10^{-2}</td>
<td>6.1×10^{-1}</td>
<td>18</td>
<td>1.2</td>
<td>2.0×10^{-5}</td>
<td>6.0×10^{-3}</td>
</tr>
<tr>
<td>8</td>
<td>0.2</td>
<td>2.0×10^{-2}</td>
<td>3.0×10^{-1}</td>
<td>19</td>
<td>1.3</td>
<td>1.5×10^{-5}</td>
<td>4.0×10^{-3}</td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td>1.0×10^{-2}</td>
<td>3.0×10^{-1}</td>
<td>20</td>
<td>1.35</td>
<td>1.0×10^{-5}</td>
<td>1.5×10^{-3}</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
<td>0.3×10^{-2}</td>
<td>2.8×10^{-1}</td>
<td>21</td>
<td>1.4</td>
<td>7.0×10^{-6}</td>
<td>1.1×10^{-3}</td>
</tr>
<tr>
<td>11</td>
<td>0.5</td>
<td>7.0×10^{-3}</td>
<td>1.8×10^{-1}</td>
<td>1.5</td>
<td>4.5×10^{-6}</td>
<td>0.8×10^{-3}</td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.18: Turbo code BER simulation with large block size and Optimized interleavers type

Figure 4.19: Turbo code FER simulation with large block size and Optimized interleavers Type
Table 4.11: Interleavers Parameter Design for a 1024-bit Frame Size

<table>
<thead>
<tr>
<th>Interleaver</th>
<th>Parameter</th>
<th>Size</th>
<th>Termination</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>32X32</td>
<td>3072,1024</td>
<td>None</td>
<td>a</td>
</tr>
<tr>
<td>Rectangular</td>
<td>21X49</td>
<td>3087,1029</td>
<td>None</td>
<td>b</td>
</tr>
<tr>
<td>Helical</td>
<td>29X36</td>
<td>3138,1044</td>
<td>Simile</td>
<td>c</td>
</tr>
<tr>
<td>Berrou Glavieux</td>
<td>32X32</td>
<td>3072,1024</td>
<td>None</td>
<td>d</td>
</tr>
<tr>
<td>Uniform</td>
<td>n/a</td>
<td>3072,1024</td>
<td>None</td>
<td>e</td>
</tr>
<tr>
<td>Uniform</td>
<td>n/a</td>
<td>3072,1024</td>
<td>JPL</td>
<td>f</td>
</tr>
<tr>
<td>Flate</td>
<td>n/a</td>
<td>3078,1024</td>
<td>Simile</td>
<td>g</td>
</tr>
<tr>
<td>Barrel shift</td>
<td>=</td>
<td>3078,1024</td>
<td>standard</td>
<td>h</td>
</tr>
<tr>
<td>Barrel shift</td>
<td>=</td>
<td>3078,1024</td>
<td>standard</td>
<td>i</td>
</tr>
<tr>
<td>Barrel shift</td>
<td>=</td>
<td>3078,1024</td>
<td>standard</td>
<td>j</td>
</tr>
<tr>
<td>One time pad</td>
<td>Terminated type</td>
<td>3078,1024</td>
<td>standard</td>
<td>k</td>
</tr>
<tr>
<td>One time pad</td>
<td>Terminated type</td>
<td>3078,1024</td>
<td>standard</td>
<td>l</td>
</tr>
<tr>
<td>Simulated</td>
<td></td>
<td>3072,1024</td>
<td>None</td>
<td>m</td>
</tr>
</tbody>
</table>

Other energy functions from the origin of IODS points based on the radial distance have also been tried, with very similar results. It is seen that the combinatorial restrictions imposed on the interleaver structure in coding scheme do not allow significantly better interleavers design to be constructed. If there is anything further to be gained in interleaver design for Turbo codes. The other factors need to be considered such as puncturing of coding, higher-order distance statistics in coding scheme.

4.7 Interleaver Design for Small Frame Size

4.7.1 Basic
In this chapter we have consider the interleaver design problem for small frame sizes. The effect of many trellis termination in coding scheme is observed by comparing the performance of the uniform interleaver type with and without termination scheme of coding. We also analyze the performance of different termination in coding schemes when used with an optimized interleaver type in turbo coding.

4.7.2 Performance Reference in Coding
As in previous Chapter we have considered ourselves to unpunctured rate of 1/3 symmetric Turbo codes with help of encoder memory $\nu = 2$ and generator $(1; 5/7)$. It is clear that we have chosen a very small block size $\tau = 64$ to analyze the effect of termination in turbo coding where it should be most pronounced scheme. The effect of interleaver choice for Turbo codes with component code and block size are indicative of what can be expected with other good component codes and similarly small block sizes. We simulate the uniform interleaver with and without termination scheme in turbo coding. To terminate the uniform interleaver in coding we use the scheme proposed by the JPL team in coding scheme. The Bit Error Rate (BER) and Frame Error Rate (FER) results for these two Turbo codes (A and B in Table 4.15) are shown respectively in Figs. 4.20; all simulations are performed by using 10 iterations with a target tolerance of $\pm 10\%$ at a confidence level of 95%. The performance of the two codes in coding scheme is almost identical at low signal to noise ratio (SNR). The trellis termination improves the performance of Turbo coding so codes at high SNR with encoder memory of $\nu = 2$. It is scene that, the frame error rate (FER) performance is better in this type of coding and the bit error rate (BER) performance is also improved in turbo coding.

### Table 4.12: small block size and Uniform interleaver

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>$6.0 \times 10^{-2}$</td>
<td>$5.0 \times 10^{-1}$</td>
<td>5</td>
<td>2.5</td>
<td>$1.0 \times 10^{-3}$</td>
<td>$1.0 \times 10^{-3}$</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>$2.0 \times 10^{-2}$</td>
<td>$3.0 \times 10^{-1}$</td>
<td>6</td>
<td>3.0</td>
<td>$3.0 \times 10^{-4}$</td>
<td>$5.0 \times 10^{-3}$</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>$9.0 \times 10^{-3}$</td>
<td>$9.0 \times 10^{-2}$</td>
<td>7</td>
<td>3.5</td>
<td>$7.0 \times 10^{-5}$</td>
<td>$0.8 \times 10^{-3}$</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>$3.0 \times 10^{-3}$</td>
<td>$4.0 \times 10^{-2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.7.3 Deterministic Interleavers Design

The performance of Turbo codes with Square, Berrou-Glavieux, and Helical interleaver types are shown in Figs. 4.22 and 4.23. It is represented by codes C, D, and E in Table 4.15. For comparison, the performance of the terminated uniform interleaver of code B is also shown in the graphs. The Helical interleaver satisfies all necessary restrictions. It is mentioned below

(a) $R$ and $C$ are relatively prime.
(b) $C$ is a multiple of $\nu + 1$ and the RSC code’s feedback polynomial is full. It makes the interleaver simile.
(c) If $C$ is even, so that the interleaver is odd-even type. This allows the same interleaver to be fairly compared with other such interleavers in a study of punctured Turbo codes in coding system.
As Turbo codes with large interleavers type, the Square interleaver performs worst performance and the Berrou-Glavieux interleaver performs better performance than the uniform interleaver design. In this case, however, the helical interleaver performs better performance than the uniform interleaver both in terms of Bit Error Rate and Frame Error Rate.

Table 4.13: small block size and Deterministic interleaver

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>6.0×10^-2</td>
<td>5.0×10^-1</td>
<td>5</td>
<td>2.5</td>
<td>3.0×10^-4</td>
<td>8.0×10^-3</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>2.0×10^-2</td>
<td>0.9×10^-1</td>
<td>6</td>
<td>3.0</td>
<td>1.5×10^-4</td>
<td>3.0×10^-3</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>9.0×10^-3</td>
<td>8.0×10^-2</td>
<td>7</td>
<td>3.5</td>
<td>3.0×10^-5</td>
<td>8×10^-4</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>2.5×10^-3</td>
<td>3.0×10^-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.22: Turbo code BER simulation with small block size and Deterministic interleavers Design

Figure 4.23: Turbo code FER simulation with small block size and Deterministic interleavers Design
Table 4.14 small block size and Optimized interleavers design

<table>
<thead>
<tr>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
<th>S No</th>
<th>Eb/No</th>
<th>BER</th>
<th>FER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>6.0×10^{-2}</td>
<td>5.0×10^{-1}</td>
<td>5</td>
<td>2.5</td>
<td>4.0×10^{-4}</td>
<td>6.0×10^{-3}</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>2.0×10^{-2}</td>
<td>0.9×10^{-1}</td>
<td>6</td>
<td>3.0</td>
<td>8.0×10^{-5}</td>
<td>0.5×10^{-3}</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>8.0×10^{-3}</td>
<td>7.0×10^{-2}</td>
<td>7</td>
<td>3.5</td>
<td>0.5×10^{-5}</td>
<td>3×10^{-4}</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>2.5×10^{-3}</td>
<td>3.0×10^{-2}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.24: Turbo code BER simulation with small block size and Optimized interleavers design
4.7.4 Optimized Interleaver Design

The performance of Turbo codes with interleavers designed by using Simulated Annealing are shown in Figs. 4.24 and 4.25. It is represented encodes F, G, and H. For comparison purpose, codes using the terminated uniform interleaver design with code B and the Berrou-Glavieux interleaver with code D are also shown in figure. We have seen from the graphs the optimized interleaver improves both the Bit Error Rate and Frame Error Rate performance in turbo coding. But in the optimized interleaver design, the effect of termination in coding becomes negligible, when the JPL termination scheme is used with a non-terminating interleaver with code G and also when a self-terminating interleaver is designed with code H. We consider the both cases separately and it is mentioned as given below:

1. When the JPL termination scheme is used in code G and the same interleaver is also used as the one for code F. The only difference is that a further two tail bits are appended to the input and interleaved sequences. Since the BJCR algorithm is used, an unterminated sequence with code F is equivalent to a terminated sequence where no tail information is available. Thus, the information advantage of code G over code F is the complete tail
information for the input sequence in coding and the tail parity information for the interleaved sequence. However, this extra information comes at a cost of code rate reduction by a factor of \( \frac{66}{64} \). It is seen from the simulation results that the information advantage is offset by the code rate reduction for this particular interleaver. It can be seen from the uniform interleaver simulations; the average the JPL termination scheme does provide a performance improvement in coding scheme. This advantage seems to depend on the particular interleaver used in turbo coding.

2. With designing an interleaver of self-terminates, we restrict the search space\(^3\), effectively reducing the algorithm’s ability with increasing the interleaver spreading factor. For this reason, while code H has the advantage of being correctly-terminating with complete tail information for both decoders, it is not only suffers the penalty of code rate reduction, but also a small reduction in spread in coding scheme.

<table>
<thead>
<tr>
<th>Interleaver</th>
<th>Parameter</th>
<th>Size</th>
<th>Termination</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>n/a</td>
<td>192,64</td>
<td>none</td>
<td>a</td>
</tr>
<tr>
<td>Uniform</td>
<td>n/a</td>
<td>198,64</td>
<td>JPL</td>
<td>b</td>
</tr>
<tr>
<td>Square</td>
<td>8X8</td>
<td>192,64</td>
<td>none</td>
<td>c</td>
</tr>
<tr>
<td>Berrou Glavieux</td>
<td>8X8</td>
<td>192,64</td>
<td>none</td>
<td>d</td>
</tr>
<tr>
<td>Helical</td>
<td>11X6</td>
<td>204,66</td>
<td>simile</td>
<td>e</td>
</tr>
<tr>
<td>Simulated Annealing</td>
<td>In Text</td>
<td>192,64</td>
<td>none</td>
<td>f</td>
</tr>
<tr>
<td>Simulated Annealing</td>
<td>In Text</td>
<td>198,64</td>
<td>JPL</td>
<td>g</td>
</tr>
<tr>
<td>Simulated Annealing</td>
<td>In Text</td>
<td>198,64</td>
<td>self</td>
<td>h</td>
</tr>
</tbody>
</table>

It is conclude that termination is beneficial for the average interleaver but its effect on optimized interleavers is negligible and it is not always beneficial in turbo coding scheme. It is considered that designing an interleaver without termination is simpler and there is also a minor gain in code rate of turbo code. It concludes that the design of non-terminating interleavers should be preferred for all block sizes in turbo code.

### 4.7.5 Two Dimensional single parity code

Message Bit 1001
Transmitted bit sequence

<table>
<thead>
<tr>
<th></th>
<th>D1=1</th>
<th>D2=0</th>
<th>P12=1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D3=0</td>
<td>D4=1</td>
<td>P34=1</td>
</tr>
<tr>
<td></td>
<td>P13=1</td>
<td>P24=1</td>
<td></td>
</tr>
</tbody>
</table>

Assume Received sequence:

\[ 0.75, 0.05, 0.10, 0.15, 1.25, 1.0, 3.0, 0.5 \quad = x_1, x_2, x_3, x_4, x_{12}, x_{34}, x_{13}, x_{24} \]

\[ L_c(x), L_c(x_{ij}) = (1.5, 0.1, 0.2, 0.3, 2.5, 2, 6, 1) \]

<table>
<thead>
<tr>
<th></th>
<th>( L_c(x_1) = 1.5 )</th>
<th></th>
<th>( L_c(x_2) = 0.1 )</th>
<th></th>
<th>( L_c(x_{12}) = 1.25 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( L_c(x_3) = 0.2 )</td>
<td></td>
<td>( L_c(x_4) = 0.3 )</td>
<td></td>
<td>( L_c(x_{34}) = 1.0 )</td>
</tr>
<tr>
<td></td>
<td>( L_c(1,3) = 3.0 )</td>
<td></td>
<td></td>
<td></td>
<td>( L_c(24) = 0.5 )</td>
</tr>
</tbody>
</table>

\[ L_{eh}(d_1) = -0.1 \text{ (new)} \]
\[ L_{eh}(d_2) = -1.5 \text{ (new)} \]
\[ L_{eh}(d_3) = -0.3 \text{ (new)} \]
\[ L_{eh}(d_4) = -0.2 \text{ (new)} \]
\[ L_{ev}(d_1) = 0.1 \text{ (new)} \]
\[ L_{ev}(d_2) = -0.1 \text{ (new)} \]
\[ L_{ev}(d_3) = -1.4 \text{ (new)} \]
\[ L_{ev}(d_4) = 1.0 \text{ (new)} \]

Original \( L_c(x_k) \)
First Iteration (first horizontal decoding)

\[
\begin{array}{|c|c|}
\hline
D_1 & D_2 \\
\hline
-0.1 & -1.5 \\
\hline
D_3 & D_4 \\
\hline
-0.3 & -0.2 \\
\hline
\end{array}
\]

Low confidence in \(D_3\) & \(D_4\)

First vertical decoding

\[
\begin{array}{|c|c|}
\hline
D_1 & D_2 \\
\hline
0.1 & -0.1 \\
\hline
D_3 & D_4 \\
\hline
-1.4 & 1.0 \\
\hline
\end{array}
\]

High confidence in \(D_1\) & \(D_4\)

Improved LLR due to \(L_{eh}(d)\)

\[
\begin{array}{|c|c|}
\hline
D_1 & D_2 \\
\hline
1.4 & -1.4 \\
\hline
D_3 & D_4 \\
\hline
-0.1 & 0.1 \\
\hline
\end{array}
\]

Improved LLR due to \(L_{eh}(d) + L_{ev}(d)\)

\[
\begin{array}{|c|c|}
\hline
D_1 & D_2 \\
\hline
1.5 & -1.5 \\
\hline
D_3 & D_4 \\
\hline
-1.5 & 1.1 \\
\hline
\end{array}
\]

\(L_{eh}(d_1) = 0\) (new)
\(L_{eh}(d_2) = -1.6\) (new)
\(L_{eh}(d_3) = -1.3\) (new)
\(L_{eh}(d_4) = 1.2\) (new)
$L_{\text{ev}}(d_1) = 1.1$ (new)  
$L_{\text{ev}}(d_2) = -1.0$ (new)  
$L_{\text{ev}}(d_3) = -1.5$ (new)  
$L_{\text{ev}}(d_4) = 1.0$ (new)

Original $L_c(x)$

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{\text{ev}}(d)$ after second vertical decoding</td>
<td>1.5</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>$L_{\text{eh}}(d)$ after second horizontal decoding</td>
<td>0.0</td>
<td>-1.6</td>
<td>-1.3</td>
<td>1.2</td>
</tr>
</tbody>
</table>

Soft output is $L(d) = L_c(x) + L_{\text{eh}}(d) + L_{\text{ev}}(d)$

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L(d)$</td>
<td>2.6</td>
<td>-2.5</td>
<td>-2.6</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Level of confidence about these decisions is high
The original message 1001 is obtained.

### 4.7.5.1 Improvement due to coding

If $E_b/N_o$ is given 14 db, coding gives better performance. If $E_b/N_o = 10$ db then coding gives no error performance. There is degradation. The cross over occurs due to every code. System has some fixed error correcting capability. If there are more errors within the block than the code is capable of correcting. System will perform poorly. Imagine that $E_b/N_o$ is continuously reduced. Then output of demodulator gives more and more errors. By this reason, some threshold to be reached where decoder becomes over whelmed with error and redundant bit consuming energy but giving back nothing beneficial in return. Then we use a class of powerful codes called turbo codes that provide error performance improvement at low value of $E_b/N_o$. The threshold point lower for turbo codes compared to convolution codes.

In BPSK, code (15, 11), Block length =15 bit, $P_r/N_o = 43776$, $R = 4800$ bit/S

**Without coding**

$$E_b/N_o = (P_r/N_o R) = 43776/4800 = 9.12 = 9.6 \text{ db}$$

$$P_b = p_u = Q(2E_b/N_o)^{1/2} = Q(2 \times 9.12)^{1/2} = 1.02 \times 10^{-5}$$

$$P_{cm} = 1 - (1 - p_u)^{11} = 1.12 \times 10^{-4} \text{ (1 error out of 11 bit)}$$

**With coding**

$$R_c = 15/11 \text{ data bit rate } =15 \times 4800/11 = 6545 \text{ bps}$$

$$E_c/N_o = P_r/(N_o R_c) = 43776/6545 = 6.69 = 8.3 \text{ db}$$

$$P_c = Q(2E_c/N_o)^{1/2} = Q(2 \times 6.69)^{1/2} = 1.38 \times 10^{-4}$$

Error rate $P_m = \binom{15}{2} p_c^2 (1-p)^{13} = 1.94 \times 10^{-6}$

Improvement Due to Coding $= 112 \times 10^{-4} / 1.94 \times 10^{-6} = 58 \text{ times}$
4.7.6 Forward error-correction scheme for third-generation high-speed wireless data services.

Turbo codes play a major role in the error channel coding scheme in wireless communication. Turbo codes emerged in 1993 and since this year it becomes a popular area of communications research due to their performances, turbo codes are being accepted as standard by many organization such as CCCDS to be used in satellite channel. In next era of wireless communications, mainly the 3G and 4G applications, there is a need to provide the best QOS (Quality of Service) provisioning. The fourth generation (4G) cellular systems are aimed at supporting the various applications of coding like voice, picture, data and multimedia over packet switched networks.

Parity bit
Puncturing

Fig. 4.26 Turbo code encoder puncturing block diagram.

Different applications have varied QOS (quality of system) requirements in terms of the data rate, bit error rate (BER), frame size and the packet error rate. Turbo codes is a most adaptable error coding scheme that is used to adapt to the varying quality of service (QOS) requirement. Hence there is a necessary to determine the performance of Turbo coding by varying all the parameters in coding which can be made more adaptable. Based on the analysis of turbo coding, the most suitable parameters are chosen. In this article we have analyze the behavior of Turbo codes for various interleaver size and structure in coding scheme. The simulation is performed for the different speed of vehicles and the graph is plotted against the bit error rate and the signal to noise ratio.

Information bit
One of the major challenges for the third-generation wireless system is to improve spectral efficiency to the extent possible. Fortunately for CDMA, the lower power requirement leads to lower interference from each user imposed on the system, which translates directly into capacity gain. Turbo codes, capable of near Shannon limit power efficiency, have recently been adopted to improve the system capacity effectively for the third-generation high-speed wireless data services by the standards-setting organizations in the United States, Europe, and Asia. In this paper, we show the performance of turbo codes under high-speed data transmission conditions and compare it with other coding alternatives. The standardized turbo codes for the cdma2000 standard, as proposed by the Telephone Industry Association in the United States, and for the UTRA/W-CDMA standard, as proposed by the European Telecommunications Standards Institute and the Association of Radio Industry and Business in Europe and Japan, respectively. This section also discusses how a high-performance, formula-based turbo interleaver (that requires storing only a small number of parameters) for any frame size has been designed. Section second explains why the characteristics of the third-generation high-speed data systems work in favor of turbo coding over convolutional coding. It also shows the performance of turbo codes under realistic cdma2000 and UTRA/W-CDMA channel and receiver models. In this section, we also showed that turbo decoding can be made very robust against internal decoder variables. ITU’s goal is to achieve a harmonized third-generation wireless standard that would allow users to roam anywhere in the world without resorting to multimode terminals. Although a small part of the overall system in coding scheme, the turbo coding specifications for cdma2000 and UTRA/W-CDMA system are designed to have as much commonality as possible toward achieving this type of goal.
We can compare the turbo code and convolution code for third generation wireless system and their performance in cdma. Is is given as below.

There are several reasons why turbo codes are especially suited for high-speed data services of third-generation wireless systems. First, at high speeds, the sufficiently long block of data in coding may be accumulated within a frame of 10 or 20 ms without any causing substantial delay in the system of coding scheme. Turbo codes become more and more effective as the block (turbo interleaver) size increases because of spectral thinning (i.e., the multiplicity of “neighbor” codeword’s becomes smaller as the interleaver size gets larger) [2]–[4]. As an example, Fig. 4 shows the performance of turbo code of rate 1/2, 1/3, and 1/4 with increasing frame sizes in comparison with convolutional codes. In each case, eight iterations are used in the decoder. Around a bit error rate (BER) of 10, for the most practical interleaver lengths that have been simulated, there is about 0.2 dB to be gained when the interleaver length is doubled.
each time. Of course, this gain eventually diminishes as the interleaver length is increased outside this range.

![Image](image1.png)

**Fig.4.28 Performance of rate 1/3 turbo and convolutional codes with power control**

It is 76.8 kbps, 20 ms frame duration, CDMA receiver model with vehicle speed of 120 kph. The second reason why turbo codes are particularly suitable for third-generation high-speed data services is that error-free data transmission is typically accomplished by an automatic repeat request (ARQ) protocol implemented in higher layers. As such, the more appropriate figure of merit is frame error rate (FER), rather than bit error rate (BER). The performance difference between turbo coding and convolutional coding becomes even larger as compared in terms of frame error rate (FER) as opposed to BER [7].

![Image](image2.png)
Fig 4.29 Interleaver Length 512 bits

Comparison of rate 1/2 turbo codes and convolutional codes over Rayleigh fading channel, no power control, \( v = 30 \) km ph. Interleaver length for 512 bits and 3072 bits.

Fig 4.30 Interleaver Length 3072bits

Fig 4.31 Rate 1/3
It is BER comparison of turbo codes with eight iterations and convolutional codes, AWGN channel. (Rate 1/3, rate 1/2, and rate 1/4). Furthermore, one notes that as the information frame size increases from 512 to 3072, the frame error rate for turbo codes decreases sharply, at least in the waterfall region, while that of the convolutional code increases significantly. Ignoring edge effects, the bit error rate for convolutional codes is essentially uniform across the frame and is a constant independent of frame size. Thus, for a given Eb/No, the expected number of bit errors increases with frame size, and the frame error rate worsens. For turbo codes system, however, the power of the coding scheme increases significantly as the frame size of code increases due to spectral thinning of component. This increase in power is more than sufficient to overcome the burden of protecting a larger frame of data. It should be pointed out that this result applies in the waterfall region it is well known that there is no
interleaver gain for turbo codes in terms of FER in the error asymptote region. For an analytic investigation of coding, the waterfall region performance of turbo codes, see [12]. The third reason for turbo codes to become an effective forward error-control technique for third-generation wireless systems is the fast power control employed in these systems. Indeed, without any power control in coding, the performance of turbo codes over convolutional code in coding scheme decreases considerably in system. As an example, shows the performance of rate $\frac{1}{2}$ turbo code and convolutional codes over a Rayleigh fading channel scheme without power control at a 30-kmph vehicle speed. As usual, eight iterations are used in the turbo decoder system. In fact, especially for the 512-bit frame size, one might argue that the BER and FER performance of the turbo and convolutional codes are so similar that the extra complexity of the turbo decoder is not worthwhile. On the other hand, shows that the use of fast power control can restore the performance advantage of turbo codes to gains close to that achieved on the additive white Gaussian noise (AWGN) channel. For this comparison, the Vehicular Test Environment Channel a specified by ITU is used as the fading channel model. In this model, there are six Rayleigh fading paths of varying delay and strength, as shown in Table II. The chip rate is 3.6864 Mchips/s, there are six samples per chip, the data rate is 76.8 kbps, and the frame duration is 20 ms. A fixed low-pass infinite impulse response filter given by $y(n) = x(n)/768 + 767x(n-1)/768$ is used for phase estimation at all vehicle speeds.

Table 4.17 Vehicular test environment channel a tapped delay line parameters

<table>
<thead>
<tr>
<th>S No</th>
<th>Taps</th>
<th>Average power in db</th>
<th>Delay samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-1</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>-9</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-10</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>-15</td>
<td>38</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>-20</td>
<td>56</td>
</tr>
</tbody>
</table>

The signal-to-noise ratio (SNR) estimation required for turbo decoding is readily obtained from the power-control algorithm. Each power-control bit is sent every 1.25 ms with a delay of 0.715 ms and an error rate of 4%. To simulate real systems accurately, interference from 20 other users is obtained by actually encoding their bits and sending them over the channel. A pulse shaping filter at the transmitter side and a matched filter at the receiver side is also
implemented in structure. Eight turbo decoder iterations are performed. Turbo codes provide substantial gain in all cases, although the cases with higher vehicle speeds and lower other-cell interference seem to favor turbo codes more than convolutional codes. For a UTRA /Wireless-CDMA (3G) third-generation wireless system, serial and parallel concatenated convolutional codes were investigated as candidates for high-speed communication, high-quality data transmission (BER = 10^{-6}).

4.7.7 The effect of interleaved parameter on Turbo Codes

In this paper we study the effect of random interleaver parameter on turbo code. The Turbo code is constructed by the random interleaver. It increases the bit error rate performance of the Turbo code. The effect of BCH code on frame error rate is also shown in this paper. The outer code of the BCH improves the iterative decoding algorithms by help of stopping method. In this method we obtain codes with performance of 0.6 db at frame error rate of 10^{-6}. Index Terms- Convolution code, BCH code, Turbo Code, Bit error rate, frame error rate.

Introduction-The concept of Turbo code was first purpose by Berrou,Glavieux and Thitimajahima in 1993.it is first introduced in paper [1] and it achieved the BER in order of 10^{-5} with SNR of 1 db of Shannon limit. The bit error rate performance is about of 1 db of Shannon limits. The basic Turbo code consists by two recursive systematic convolution code and one systematic code. We use three variation of inner Turbo code with information block length of 16384, sixteen state rate half convolution component codes. The overall rate is 1/3. There are two RSC (Recursive systematic convolution) code and it is represented by Berrou component code and Primitive feedback polynomial component code.

\[ G_1(D) = \frac{1}{(1+D^8)(1+D^2+D^3+D^4)} \]

\[ G_2(D) = \frac{1}{(1+D+D^2+D^4)(1+D^3+D^5)} \]

\[ G_1(D) \text{ and } G_2(D) \text{ is known as Berrou component code and primitive feedback polynomial component code. The corresponding turbo code is obtained by concatenation of these two codes.} \]

We discuss the several alternatives to improve the bit error rate and frame error rate performance. There are some deficiencies in iterative APP decoder .These deficiencies will be discussed here. We will discuss the some positive approach of Turbo code. Finally, we present conclusion of overall bit error rate performance.

**Table 4.18 Iteration and BER at \( E_b/N_0=0.4 \text{ db} \)**
There are two small error regions in which error is lowest. These error region occur in between 9 to 17 iteration and from 35 to 40 iteration. Tease region is known as region R1 and R2. The turbo code uses either a randomly chosen interleaver or a deterministic interleaver. The number of bits errors in an incorrect frame is relatively small. One of the best purposed methods to lower the error was concatenated with high rate algebraic outer code. It was observed that the location in the information block that was affected by incorrectly decoded frame in the error region is not random. It has small subset of the possible error in this region. There is little additional way to lower the error region. We use the random interleaver or outer block code to reduce the small error region. Thus, we prefer the method that fixes the error region without the need to more examine the interleaver and component code. The lowering the error by factor of 10 was a matter of dealing with small number of codeword’s with small weight. The scheme that removes the error can greatly improve the FER performance of turbo codes operating at low signal to noise ratio. In this paper we investigate the effect of removing the error regions by use of random interleaver and codes.

Improvement in frame error rate

The table can be divided into three parts. The first region in which table shows no improvement in frame error rate in first region from 5 to 10 iterations. The second region 11 to 20 iteration of table shows a linear improvement on log scale and third resign from 21 to

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Information bit in error</th>
<th>Iteration</th>
<th>Information bit in error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11000</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>10800</td>
<td>19</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>10500</td>
<td>20</td>
<td>300</td>
</tr>
<tr>
<td>4</td>
<td>10100</td>
<td>25</td>
<td>2000</td>
</tr>
<tr>
<td>5</td>
<td>900</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>650</td>
<td>35</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>300</td>
<td>36</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>90</td>
<td>37</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>38</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>40</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>45</td>
<td>1600</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>46</td>
<td>800</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
<td>47</td>
<td>200</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>48</td>
<td>50</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>49</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>9</td>
<td>50</td>
<td>1</td>
</tr>
</tbody>
</table>
28iteration shows the saturation in improvement of the FIR with increasing the number of iterations. This is called as small number of error. We will see that an outer code lower the error.

Table 4.19   Iteration and FER at $E_b/N_0=0.4$ db

<table>
<thead>
<tr>
<th>No of Iteration</th>
<th>FER</th>
<th>No of Iteration</th>
<th>FER</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$10^{-00}$</td>
<td>17</td>
<td>$6\times10^{-3}$</td>
</tr>
<tr>
<td>6</td>
<td>$10^{-00}$</td>
<td>18</td>
<td>$4\times10^{-3}$</td>
</tr>
<tr>
<td>7</td>
<td>$10^{-00}$</td>
<td>19</td>
<td>$2\times10^{-3}$</td>
</tr>
<tr>
<td>8</td>
<td>$10^{-00}$</td>
<td>20</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>9</td>
<td>$10^{-00}$</td>
<td>21</td>
<td>$8\times10^{-4}$</td>
</tr>
<tr>
<td>10</td>
<td>$10^{-01}$</td>
<td>22</td>
<td>$6\times10^{-4}$</td>
</tr>
<tr>
<td>11</td>
<td>$8\times10^{-1}$</td>
<td>23</td>
<td>$4\times10^{-4}$</td>
</tr>
<tr>
<td>12</td>
<td>$6\times10^{-1}$</td>
<td>24</td>
<td>$4\times10^{-4}$</td>
</tr>
<tr>
<td>13</td>
<td>$4\times10^{-1}$</td>
<td>25</td>
<td>$4\times10^{-4}$</td>
</tr>
<tr>
<td>14</td>
<td>$2\times10^{-1}$</td>
<td>26</td>
<td>$4\times10^{-4}$</td>
</tr>
<tr>
<td>15</td>
<td>$10^{-4}$</td>
<td>27</td>
<td>$4\times10^{-4}$</td>
</tr>
<tr>
<td>16</td>
<td>$8\times10^{-4}$</td>
<td>28</td>
<td>$4\times10^{-4}$</td>
</tr>
</tbody>
</table>

It is observed that no frames converged to moderate number of errors. In other words, one of the following three possibilities has occurred.

(1) A small number of errors

(2) Moderate number of error

(3) Large number of error

The iterative APP decoder exhibits either large number of errors or a small number of errors. It is seen that this is similar to decoding of long convolution code with sequential decoding. The iterative decoding is not maximum likelihood for the overall Turbo code the most of frames that converge to a small number of errors have maximum likelihood decoding. It is
observed that, in some cases, although the number of errors is small, but the decision is not a maximum likelihood decision.

Improvement in frame error rate by help of concatenation code
The concatenated turbo code is applied only for weak bits. It is obtained with help of outer code. Let us define the BCH code with parameters \((n, k, t)\), where \(n\) is the block length, \(k\) is the information length and \(t\) is the number of errors to be corrected by the algebraic decoder.

The BCH code is used to effectively stop the iterative inner decoder by checking after each iteration when outer decoder is successful, the iterative inner decoder is stopped. This enables the decoder to stop decoding those frames with an oscillating number of errors, as long as minimum numbers of error falls within decoding radius of BCH code. By increasing the maximum numbers of error and correctly processing frames improves the BER performance.

The probability of undetected errors after iteration is given by

\[
P(B) = P(B|A) \cdot P(A) + P(B|A') \cdot P(A')
\]

\[
P(A) < P(B|A)
\]

An error pattern that results from the hard decision output of the inner decoder will typically have a small number of error that a random chosen error pattern. Thus, the hard decision output will lie in the local substance of the complete space centered at the correct codeword. It is a reasonable to assume, however, that the density of BCH code wards within the local substance is approximately the as the density of codeword’s in complete n tuple space. The probability is that randomly chosen error pattern with more than \(t\) errors lies within a radius of an incorrect codeword. We note that the hard decision output error pattern of the inner decoder is not completely random. The distribution of the outer BCH code words is independent of the distribution of inner code words.

The undetected error probability of concatenated system is obtained by multiply the number of iteration into undetected error probability. We have verified by simulation method that the undetected probability is very close to the predicted by the equation of undetected probability for the rate 1/3, length of block of 1024 including 4 tail bits \((1020,950, t=7)\). The result of probability of error is compared with help simulation method. The no error is obtained at block length=1024, \(t=7\), \(E_b/N_0 =1.03\) db in simulation method. The error probability of undetected error \(P_u 1.29 \times 10^{-6}\) is obtained with help of theoretical method in coding scheme.

There are 235711 frames are simulated at block length of 1024.
Undetected Error probability for block length 1024 is given below.

Table 4.20 Undetected Error probabilities

<table>
<thead>
<tr>
<th>S No</th>
<th>Time t</th>
<th>$E_b/N_0$ (db)</th>
<th>$P_u$</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>6</td>
<td>1.03</td>
<td>$1.3 \times 10^{-6}$</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>7</td>
<td>0.87</td>
<td>$1.9 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

Since $P_u$ is very small. It does not affect the FER. If $t$ is smaller than $t$, then $P_u$ will be very small for large code length.

Simulation results

It is clear that asymmetric turbo codes give the good bit error rate and frame error rate performance in region $R_1$ and $R_2$. It is also helpful in concatenation scheme. For example, $(49152,16240)$ code concatenated with $(16380.16240.10.8)$ binary BCH outer code we uses 16 state component code, 4 tail bits added with BCH code words prior to inner encoding The average number of iteration is in order of 10 to 15.

The undetected error probability is estimated from (2) as $P_u 9.29 \times 10^{-14}$. The undetected error probability for 100 iteration per frame is about $10^{-11}$ and we expect an order of magnitude less for average of 10 iteration per frame, which is typical for SNR of interest Thus undetected probability of error is much smaller than our target FER of $10^{-6}$. We have simulate the $10^8$ frame but no error was observed. We have shown that the FER performance of the concatenated Berou code and concatenated Primitive Berrou code, both with rate 0.330 along with the frame error rate performance of the corresponding inner turbo codes alone. The improvement in performance is dramatic both in $R_1$ and $R_2$ region.

The concatenated berrou code for FER of $10^{-4}$ is best choice. But the concatenated asymmetric Berrou code is chosen best for FER of $10^{-6}$. It is clear in [19] that turbo codes are within 0.7 db of the sphere packing bound for several code rates with FER of $10^{-4}$ It is very difficult to obtained low FER at $R_1$ region. In asymmetric primitive code, the frame error rate is about $10^{-6}$ at 0.6 in $R_1$ and $R_2$ region. We have seen that the computational complexity of BCH code is small as compared to an iterative APP decoder.

Random interleaver or deterministic random interleaver gives the good BER performance in long frame length of turbo codes. It gives the certain deficiencies into the code in region $R_1$
and R2. The high rate outer code is not effective for lowering the error rate. The frame with large number of error of oscillating converges in long time. The effectiveness of an BCH code is verified in simulation. We have obtained a new code with performance of 0.6 db with FER of $10^{-6}$ db this scheme has advantage for practical system implementation.

4.7.8 Survey Analysis

(1) In IS – 95 CDMA System, BPSK, QPSK, Convolution coding with Viterbi coding is used.

(2) GSM, Band width is limited & GMSK, $W T_b = 0.3$Hz/Bit/s

(3) The benefit of interleaver improve with increased vehicle speed, above speed 20 Km/h, Interleaver is required to increase the gain at higher speed.

(4) Speed coding = 32Kbps A/D PCM is used.

(5) DECT Radio link, Cyclic redundancy coding is used.

(6) In leased Line telephone modem

<table>
<thead>
<tr>
<th>Year</th>
<th>System</th>
<th>Bit Rate (bps)</th>
<th>Signal Rate</th>
<th>Modulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1984</td>
<td>V.32</td>
<td>9600</td>
<td>2400</td>
<td>32 QAM</td>
</tr>
<tr>
<td>1991</td>
<td>V.32</td>
<td>14400</td>
<td>2400</td>
<td>128 QAM</td>
</tr>
<tr>
<td>1994</td>
<td>V.34</td>
<td>28800</td>
<td>3429</td>
<td>960 QAM</td>
</tr>
<tr>
<td>1996</td>
<td>V.34</td>
<td>33600</td>
<td>3429</td>
<td>1164 QAM</td>
</tr>
<tr>
<td>1998</td>
<td>4.90</td>
<td>56000-33600</td>
<td>8000</td>
<td>PCM (M-PAM)</td>
</tr>
<tr>
<td>2000</td>
<td>V.92</td>
<td>56000-48000</td>
<td>8000</td>
<td>Trellis Coded PCM</td>
</tr>
</tbody>
</table>

(7) Personal access coding system, speech coding, and 16 bit ADPCM

(8) Digital Cordless System

Table 4.22 Bit rate and modulation for system

<table>
<thead>
<tr>
<th>System</th>
<th>Bit Rate (bps)</th>
<th>Modulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT2</td>
<td>32000</td>
<td>GFSK</td>
</tr>
<tr>
<td>DECT</td>
<td>32000</td>
<td>GFSK</td>
</tr>
<tr>
<td>PHS</td>
<td>32000</td>
<td>$\pi/4$ DQPSK</td>
</tr>
<tr>
<td>PACS</td>
<td>32000</td>
<td>$\pi/4$ QPSK</td>
</tr>
<tr>
<td>IS -</td>
<td>-</td>
<td>$\pi/4$ DQPSK</td>
</tr>
</tbody>
</table>
4.7.9 VHDL Programming of State Diagram Representation

The code for the 3 bit shift registers with C1 & C2 outputs and the waveforms for the various inputs.

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY reg2 IS
    GENERIC( N : INTEGER :=3);
    PORT( D : IN STD_LOGIC;
          clock : IN STD_LOGIC;
          c1,c2 : OUT STD_LOGIC;
          Q : buffer STD_LOGIC_VECTOR(0 TO N-1));
END reg2;
ARCHITECTURE behavior OF reg2 IS
    SIGNAL x : STD_LOGIC;
    BEGIN
    PROCESS(clock)
    BEGIN
    WAIT UNTIL clock'EVENT AND clock='1';
    Q(0)<=D;
    Q(1)<=Q(0);
    Q(2)<=Q(1);
    x   <=  Q(0) XOR Q(1);
    c1  <=  Q(2) XOR x;
    c2  <=  Q(0) XOR Q(2);
    END PROCESS;

END behavior;

<table>
<thead>
<tr>
<th>D</th>
<th>clock</th>
<th>c2</th>
<th>c1</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>clock</th>
<th>c2</th>
<th>c1</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>clock</th>
<th>c2</th>
<th>c1</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>clock</th>
<th>c2</th>
<th>c1</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
Data input $D = \ "11001\"$.

Data input $D = \ "00101\"$.

Data input $D = \ "10101\"$.

Data input $D = \ "01010\"$.
Data input

D = "11101".

Data input D = "00111"

Data Input D = "11100".
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY reg4 IS
    PORT(clock, data : IN STD_LOGIC;
         z : OUT STD_LOGIC);
END reg4;

ARCHITECTURE behavior OF reg4 IS
    TYPE state_type IS(A,B,C,D);

    Data input D = “00100”.

    Data input D = “01110”

    Data input D = “01110”

    Fig 4.34 The state machine coding for the 3 bit shift register
SIGNAL y : state_type;
BEGIN
PROCESS(clock)
BEGIN
    IF(clock'EVENT AND clock='1')THEN
        CASE y IS
            WHEN A=>
                IF data='0'THEN
                    y <= A;
                ELSE
                    y <= B;
                END IF;
            WHEN B=>
                IF data='0'THEN
                    y <= C;
                ELSE
                    y <= D;
                END IF;
            WHEN C=>
                IF data='0'THEN
                    y <= A;
                ELSE
                    y <= B;
                END IF;
            WHEN D=>
                IF data='0'THEN
                    y <= C;
                ELSE
                    y <= D;
                END IF;
        END CASE;
    END IF;
END PROCESS;
END IF;

END PROCESS;
\[ z <= '1' \text{ WHEN } y = D \text{ ELSE '0'}; \]

END behavior;
4.7.10 Low power Consumption parameter in Turbo code

The series cs3500 of specific virtual component is introduced to deliver highest forward error correction performance in third generation mobile communication and satellite communication. The standard product of this series is scheduled in programmable logic version with optimal maximum performance and minimal power consumption. It is economic in power that has been made possible by incorporate unique signal processing algorithms. The series cs3510, cs3610 offers support for error correction standard as specified by the third generation product partnership for average data rate of 2.048 Mbps. In burst processing mode, the cs 3500 support data rates more than one order of magnitude higher than average rate. The standard 16 bit processor interface enables easy user set up of coding parameters. The series cs 3520 and cs3620 supports in cdma standards. These new product is used with next generation cellular phone in base station and mobile set. The turbo code with unique power saving techniques is incorporated here first time. Amphion is long time supplier of forward error correcting solution., notably Reed Solomon and Viterbi cores, in highly reusable firm IP(targeted net list) version for ASIC and FPGA. Cs3000 channel coding as the heater of IP based soc platforms today digital communication. The use of turbo codes enables the data transmission efficiency in digital communication systems, enabling reliable communications over power constrained communications channels at close to Shannon limit. This technique provides a particular robust error correction solution designed to combat channel fading.
Turbo codes present a more elegant and silicon efficient solution than so called concatenated FEC solution, typically assembled from reed Solo man block coder and Viterbi convolution coder elements. 16 ½ said Stephen Farsen, amphioni 16 ½ s VP engineering, Using 16 ½ Turbo codes, 3G system designers get an off the shelf, standards compliant solution that has been architect top down with more aggressive power budgets in mind 16 ½. In the Amphion Turbo codes decoder, a mechanism for early termination of iterative decoding helps conserve power by intelligent and timely suspension of processing activity. This feature contributes to power consumption saving for system on a chip designs destined for use in battery power mobile cell phone.

Turbo Decoder chip is designed by Bell lab. Which will be licensed to manufacturers of the wireless data transmission? It is powerful enough to handle the data rate up to 24 Mbps. It is nearly ten times faster than today’s most advanced mobile networks. The chip was designed during the presentation at the international solid state circuits conference here by two of the bell lab researchers who developed the chip. The high speed downlink packet access is an evolutionary enhancement to universal mobile telecommunication system spread spectrum technology, also known as wideband code division multiple access. The chip is fast enough not only to support first generation high speed data link packet access system, which will be faster enough not only to support first generation high speed downlink packet access system, which will be after transmission speed between 5 to 10 mbps, but also future multiple input / output high speed system, which are expected to achieve peak data rates up to 20 mbps.

DISCUSSION OF SYMBOL BASED TURBO CODE

5.1 Introduction

For digital wireless communication system, the main purpose of the channel code in wireless is to add redundancy bit to the binary data stream to combat the effect of signal degradation in the channel of wireless communication. Ideally, channel codes in communication system will meet the following requirements.
1. Channel codes of wireless communication system should be high rate to maximize data throughput.
2. Channel codes must have good BER performance at the desired SNR to minimize the energy needed for transmission in wireless system.
3. Channel codes must have low encoder/decoder design complexity to limit the size and cost of the transceivers of system.
4. Channel code of wireless communication should introduce only minimal delays in system, especially in voice transmission, so that no degradation in signal quality in communication is detectable. These requirements are very difficult to obtain simultaneously in communication system. Excellent performance in none requirement usually comes at the price of reduced performance in another. However, for cellular voice communication and data communication, it is very desirable that all these requirements be met, which makes cellular communication systems very difficult to design in wireless communication.

5.2 Shift Register for linear feedback Sequences

5.2.1 Shift Registers for Binary Linear Feedback

The Turbo Code encoder is designed by using Recursive Systematic Convolutional (RSC) encoders in communication. Since Recursive Systematic Convolutional (RSC) codes are formed by Linear Feedback Shift Register (LFSR) in communication. It will review the aspects of Linear Feedback Shift Register (LFSR) sequences which are relevant to Turbo Codes in wireless communication. An extensive study on shift register sequences was explained by Golomb in [11]. A binary Linear Feedback Shift Register is an arrangement of n memory elements in which each store a binary variable zero and one. The basic structure of a LFSR is given in Figure 5.1. At each step of the sequence, the value in each memory element in shift register is shifted one element to the right, and the left most memory element in shift register is calculated as a linear combination of the values in the memory elements by help of the previous step. Collectively, the values stored in all the memory elements in the registers are called the state of the Linear Feedback Shift Register. The terms Ci' is binary variables that indicate the position of the switches in figure. A 1 indicates a closed switch, and a 0 indicates an open switch. From the structure of an Linear Feedback Shift Register, it is clear that the generated sequence of aₙ satisfies the recursive equation given below.
with an initial state \((a_1, a_2, \ldots, a_n)\). The sequence generated by an Linear Feedback Shift Register will vary depending on its structure and its initial state. Any sequence \(a_n = (a_0, a_1, a_2, \ldots)\) may be described by the generating function of \(G(x)\), which for an Linear Feedback Shift Register sequence is as follows,

\[
G(x) = \sum_{i=0}^{n} a_i x^i
\]

\[
= g(x) / f(x)
\]

The polynomial \(f(x) = \sum_{i=0}^{n} c_i x^i\) is known as the characteristic polynomial of the sequence \(a_n\) and of the shift register which produced it. Note that \(f(x)\) is solely a function of the structure of the Linear Feedback Shift Register. It is independent of the initial state. The function \(f(x)\) is a manic polynomial of degree \(n\), i.e., \(c_n = c_0 = 1\).

In Linear Feedback Shift Register, the next entry in the sequence is only dependent on the current state in equation. If any particular state in sequence has occurred for the second time, the generated sequence must be periodic from that point on. Therefore, the maximum period of an Linear Feedback Shift Register sequence is \(2^n - 1\), which corresponds to one cycle through each of the \(2^n - 1\) non-zero states. These sequences are commonly referred as maximum length sequences and it is known as m-sequences. Golomb showed that the period of an Linear Feedback Shift Register sequence with characteristic polynomial \(f(x)\) is the smallest integer \(p\) such that \(f(x)\) divides \(1 - x^p\) (modulo 2\(^n\) arithmetic) \[11\]. This integer \(p\) is also known as the exponent of \(f(x)\). A necessary, but it is not sufficient condition of \(f(x)\) to produce an m sequence is that \(f(x)\) will be irreducible. The number of polynomial of degree \(n\) which have maximum exponent is given by \((2^n - 1)/n\) where \((\ )\) is the Euler function in equation.

### 5. 2.2 Generalized Linear Feedback Shift Registers
In a generalized Linear Feedback Shift Register, each memory element contains a value in \((0, 1 \ldots q - 1)\) and all calculations are performed over the finite field \(GF(q)\). A sequence, \(a_0, a_1, a_2, \ldots\) generated by a generalized Linear Feedback Shift Register satisfies the linear recurrence relation that is given below

\[ a_k = \sum_{i=0}^{n} c_i a_{k-i} \]

Where all the elements are taken from a finite field of \(GF(q)\). This equation can be rearranged by \(a_k\) in terms of a linear combination of the previous \(n\) values of the sequence \(a_k\) as follows in equation,

\[ a_k = -c_0^{-1} \sum_{i=0}^{n} c_i a_{k-i} \]

The basic structure of a generalized Linear Feedback Shift Register is shown in Figure 10.1. The generating function for the sequence produced by a generalized Linear Feedback Shift Register is \(G(x) = g(x)/f(x)\). This is the same as in the binary form, with the exception that \(f(x)\) and \(g(x)\) have main coefficients in \(GF(q)\) equation. Similar to the binary form, the period of a generalized Linear Feedback Shift Register sequence is given by the smallest integer \(p\) so that \(f(x)\) divides \(1 - x^p\) of modulo \(q\) arithmetic. The set of all possible sequences generated by an Linear Feedback Shift Register is called the solution space of \(f(x)\), it is denoted by \(S(f)\).
If a sequence, after the multiplication of each term by an element in GF(q) results in a shift of the original sequence, then this element is known as a multiplier of the sequence, and the number of positions the original sequence is shifted forward direction is known as the span of the multiplier. Furthermore, the set of sequences obtained in the equation when a given sequence is multiplied by all the non zero element of GF(q) is known as a block. Dan Laksov, in [12] states the following two theorems related about the relation between multipliers, spans and blocks.

**THEOREM 1:** The multipliers of the sequences of S(f), where function f(x) is irreducible, period p, e = GCD(p, q - 1) and = p/e, are exactly the elements of GF(q) which satisfying the $x^e = 1$, and it have the span 0,µ,2µ, ……(e-1)µ.

**THEOREM 2:** If function f(x) is irreducible, and there is t unordered sequences (i.e. Which are not shifts to each other) in each block sequence, and there are b blocks, and t = (q - 1) /e , b = (q^n -1) /( µ (q - 1)). Theorem 1 and 2 can be applied to an Linear Feedback Shift Register with irreducible polynomial f(x) with coefficients in GF(q), which produces the sequence of the maximum period q^n - 1, and q is a prime. The following results are given below.

$\mu = q^n - 1/(q - 1)$ and span 0,µ,2µ, ……..(q-2)µ, t = 1, b = 1

Therefore, the solution space S(f) consists by one block of one unordered sequence. In other words, S(f) contains $q^n - 1$ phases (or forward shifts direction) of one unique sequence. It is clear that the minimum non-zero span is $q^n -1/(q -1)$, which is equal to the period $2^n -1$ in binary case, but for larger q is less than the period of the sequence.

### 5.2.3 Convolutional Encoders

Convolutional encoder in turbo code can be divided into two main categories, the first traditional Non-Recursive Convolutional (NRC) encoder and the second Recursive Systematic Convolutional (RSC) encoder. The central component of the Non-Recursive Convolutional encoder is the shift register, which stores the previous values of the input bit stream. The outputs are obtained by linear combinations of the current and past input values. This particular encoder is nonsystematic encoder, which represents that the systematic input of data is not directly sent as an output. However Non-Recursive Convolutional encoders can be either systematic or non-systematic. The structure of the Non-Recursive Convolutional
encoder can be expressed in terms of the generator matrix \( G = [G_1(D) \ G_2(D)] \) where \( G_1(D) = 1 + D + D^2 + D^3 + D^4 \), and \( G_2(D) = 1 + D^4 \). The sequence generated by this Non-Recursive Convolutional encoder will be \([u(D)G_1(D) \ u(D)G_2(D)]\), and \( u(D) \) is the D-series representation of the input data.

The Recursive Systematic Convolutional encoder is commonly used in Turbo Codes. The Recursive Systematic Convolutional encoder contains a systematic output, and more importantly, a feedback loop. The generator matrix for the Recursive Systematic Convolutional encoder is \( G = [1 \ A(D)/B(D)] \) and \( A(D) = 1 + D^4 \), \( B(D) = 1 + D + D^2 + D^3 + D^4 \). The terms \( A(D) \) and \( B(D) \) are called as the feed-forward and feedback polynomials respectively. The feedback structure of the Recursive Systematic Convolutional encoder is encapsulated in the feedback polynomial \( B(D) \), which is the same as the characteristic polynomial of function \( f(x) \) described in previous section. Therefore, if \( B(D) \) has maximum exponent equal to \( 2^n - 1 \), then the period of the output sequence of the Recursive Systematic Convolutional encoder will be \( 2^n - 1 \) where \( n \) is the degree of \( B(D) \). The output sequence generated by this Recursive Systematic Convolutional encoder is given by \([u(D) \ u(D)A(D)/B(D)]\). Therefore, an Recursive Systematic Convolutional encoder with input sequence \( u(D) \) will produce an infinite weight output sequence, with the exception of the the sequences \( u(D) \) which are a multiple of the feedback polynomial of \( B(D) \). Note that this only includes a fraction \( 2^n \) of all the possible input sequences. Furthermore, the degree of \( u(D) \) must be at least that of \( B(D) \) [16].

### 5.2.4 Turbo Code symbol based Encoder

The traditional Turbo Code encoder is designed by concatenating two Recursive Systematic Convolutional encoders with an interleaver in between. Hence the systematic output of the second Recursive Systematic Convolutional encoder is omitted to increase the code rate. The Turbo Code encoder used in [1, 3]. Due to the interleaver between the Recursive Systematic Convolutional encoders and the nature of the decoding operation in turbo code, the Turbo Code encoder will be operate on the input data stream in N-bit blocks in coding system. Therefore, Turbo Code is known as linear block codes. The interleaver appears to play only a minor role in the
Encoder in turbo coding, in actuality the interleaver is a very important component in coding system. An estimate of the bit error rate performance of a Turbo Code can be obtained if the code word weights and code geometry is known. But we would like to avoid pairing low weight codeword’s from the upper Recursive Systematic Convolutional encoder with low-weight words from \( (Y_k^2) \).

\[ d_k \]

\[ x_k \]

\[ \text{shift register} \]

\[ y_k^1 \]

\[ \text{shift register} \]

\[ Y_k^2 \]

**Figure 5.2 Turbo Code symbol based Encoder of Rate 1/3**

Many of these undesirable low weight pairings in turbo coding can be avoided by properly designing the interleaver in system. Since the Recursive Systematic Convolutional encoders have infinite impulse response, the input sequence of code consisting of a single one will produce a high weight codeword.

The two suitably placed 1's in the input sequence of diagram can drive the Recursive Systematic Convolutional encoder out of the zero state and back again within a very short span. If the interleaver maps these sequences to similar sequences in diagram, the overall codeword weight will below. For moderate signal to noise ratio a values, the weight distribution in the first several low weight codeword’s is required. For small interleaver sizes of \( N \), the complete weight distribution of code can be archived by an exhaustive search by
using a computer. Calculating the weight distribution for large size N is a very difficult problem in coding. Once the weight distribution of coding is known, an estimate of the bit error rate may be achieved by using the union bound.

5.2.5 Design of interleaver
The main role of the interleaver in a Turbo Coding system is to change the pattern of low weight input sequence which results in a low weight output in coding after passing through the first Recursive Convolutional code so that the output weight of the second Recursive Convolutional code is very high. The main focus of designing interleaver has been on breaking weight two input sequence which drives the interleaver encoder out of the zero state in the diagram and back into the zero state with short span resulting in low weight outputs in diagram. For weight of two input sequence which cannot be broken, it is desirable that the span of such sequences are large in at least one of the Recursive Convolutional codes. The interleaver needs to permute positions near the right end of the block in coding to positions far away from the right edge so as to minimize edge effect in coding system. Consider an Recursive Convolutional encoder with transfer function \( G(D) = \frac{A(D)}{B(D)} \). but we know that if \( B(D) \) has maximum exponent, so that \( G(D) \) is a periodic sequence with period of \( p = 2^n - 1 \), here \( n \) is the degree of \( B(D) \). We know that the solution set in equation of \( G(D) \) consists of all the \( 2^n - 1 \) phases with right shifts, known as an m-sequence in system. The phases of this m sequences in the diagram correspond to the impulse response of \( G(D) \) when the input sequence, \( u(D) \), made of a single one at position of \( i \). Therefore, we label the all phases of these sequences by the integer \( i = 1, 2, \ldots, p \). It can be clear that the set of all phases of an m sequence constitute a group with binary addition. The order of each element in this type of group is equal to two, meaning that the sum of each phase in system with itself results in the all zero sequence. It is denoted as the zero phases in system.

We refer to time positions of input data block which are congruent to \( i \) modulo \( p \) as \( C_i \). \( C_i = \{t_i, t_i [1,N], t_i = i \mod(p)\} \). If the system is already represented as in phase \( a \), then an impulse response at position \( t \ C_i \) will result in phase \( b = a \ i \) at the output of the corresponding Recursive Convolutional encoder, \( \text{Exor} \) denotes the group addition of the phases. it is clear that an impulse at position \( t \ Ca \) will result in the zero phase in system. with process of interleaving, if two elements of coding within a given \( Ci \), \( i = 1,2,\ldots p \), are mapped onto two positions within a given \( C_j \), then these positions constitute a weight two, zero-phase sequence.
which remains zero-phase in diagram after interleaving. The interleaver can map the elements of a given \( C_i \), \( i = 1 \ldots p \), into different \( C_j \)'s if and only if the number of positions in coding with each \( C_i \) is less than \( p \). This in turn means that the block length \( N \) should satisfy \( N < p^2 \). It is clear that, \( N = p^2 \) and the interleaver will not be able to break all the weight two input sequences in equation. The fraction of the unbroken weight two sequences within a given \( C_i \) is at least \( 1/p \) in system. The interleaver must be designed such that the span of such unbroken sequences in system is maximized.

A. interleaver presented by Khandani, which is optimal in breaking weight of two input sequences and maximizing the span of the unbroken weight of two sequences in [18]. so interleaver is obtained by partitioning the input block into sub blocks of length \( p \) with applying a cyclic shift of \( i \) positions, \( i = 0, 1 \ldots \) to the elements of the \( i \)-th sub block. The effective number of cyclic shifts within \( i \)-th sub-block is equal to, \( i \) mod \( p \). The bit error rate performance of a Turbo Code with this type of interleaver performed poorly in system. Therefore, the bit error rate performance of a Turbo Code is not obtained solely by the weight two input sequences.

In [18], an alternative interleaver is designed by defining the distance functions of each phase and develops a linear objective function in system which is explained by using the Hungarian method [19]. These interleavers result in bit error rate performance superior to that of a random type interleaver. A good review of other non-random, 'semi-random' interleavers is performed by S. Dolinar and D. Divsalar in [20]. The authors considered a non random interleaver that ensures that the minimum distance of weight two input sequences are given as \( p^2 N \), where \( N \) is represented by block length of code. They also define as 'semi random' interleaver, which they called S random. This type of interleaver is designed by randomly selecting integers \( i, 1 \leq i \leq N \) without replacement in the system.in which each selection is compared with the previous \( S \) selections of the system. If the current selection of system is within a distance \( S \) to the previous \( S \) selections of system, then it is rejected automatically. This process has the effect of permuting all positions of within a distance of \( S \) in the original block, to a distance greater than \( S \) in the designing interleaved block. but this process is not guaranteed to complete successfully in system, Dolinar and Divsalar seen that setting \( S < \) usually produces a desirable solution within a reasonable time. Many other articles about interleaver designing were presented at the IEEE International Symposium on Turbo Codes and Related Topics in Brest France, September 1997 [21, 22, 23].
Symbol Based Turbo Code is our new method for combining Turbo Codes with different modulation techniques in the system [28]. The motivation for this combination of coding is to take advantage of inherent trade-offs between bit error rate performance, spectral efficiency, code rate, and decoder complexity in system. The main aim of our study has been to produce moderate and low rate (1/3 - 1/32) codes with good bit error rate performance without dramatically increasing the complexity of the decoder in system. These codes are suited for spread spectrum communication systems where the lower rates in system can be easily accommodated by the large spreading gains. A secondary goal will be produce spectrally efficient codes which do not suffer the same bit error rate performance degradation of traditional Turbo Codes when punctured to increase their code rates in system. Unlike the other methods reviewed in previous Chapter of this thesis which use the outputs from all the parallel data streams of the Turbo Code encoder in system to select a point in a signal space, Symbol Based Turbo Codes in system modulate each parallel data stream independently. In this chapter, we are generalized the structure of the Symbol Based Turbo Code encoder and decoder.

5.3 Symbol-Based Turbo Code

5.3.1 Encoder

The Turbo Code encoder with Symbol Based combines the traditional Turbo code encoder with different modulation schemes by using the parallel data streams into n bit symbols or sub blocks. These sub blocks are then mapped to a $2^n$ point signal set. In order to maintain the correspondence between n bit encoder output sub blocks and n bit input sub blocks, the interleaver will be restricted to operate on a sub blocks -by- sub blocks basis. This type of interleaver permutes the input data block into n-bit symbols or sub blocks, it does not change the order of the bits in symbol or sub block. With this type of restriction in place, the encoder of symbol based effectively operates over n-bit symbols instead of binary symbols in system, thus the name, Symbol Based Turbo Codes or sub block. The diagram of the Symbol-Based Turbo Code encoder is shown in Figure. The input block or symbol is parsed into n bit input symbols, $d_k$, and it is encoded by traditional Turbo Code encoder with a restricted interleaver in coding system. Every parallel data bits stream then maps into n-bit encoded symbols of $2^n$ point signal set. The over-all code rate is represented by $n/3j$ and n is the symbol size and j is the signal size. This method may be easily adapted to Turbo Codes which concatenate more
than two Recursive Systematic Convolutional encoders. Furthermore, each parallel data stream in coding system can be modulated with a different modulation scheme in system. Since, there is a great deal of flexibility in designing of codes with very high data rates as well as very low rates.

Figure 5.3: Three Stage Basic and Merged Trellis Diagrams
By changing the input data block into n-bit symbols, we are in essence compressing n sections of the encoder trellis in coding into one. Figure shows two stages of the Recursive Systematic Convolutional trellis with generator (13, 11). The solid lines and dotted lines represented by corresponding to the input bit of 0 and 1 respectively. Figure also shows the trellis diagram of coding system when two stages of the original trellis are combined together.

Similarly, Figure shows three stages of the basic Recursive Systematic Convolutional trellis and the three stage combined trellis. In general, there are $2^n$ branches in diagram leaving each state of an n-stage merged trellis diagram, which is represented by $2^n$ possible n-bit binary values.

It is clear that when the number of branches leaving each state is greater than the number of states in the combined trellis, parallel transitions occur in coding system. These parallel transitions with short error events cannot be broken by a symbol by symbol interleaver, and consequently in coding system, it degrades the performance of the coding. Therefore, the
symbol size is limited by $n^M$ and M is represented by number of memory elements in the RSC encoder. The diagram of the Symbol-Based Turbo Code encoder allows for different modulation scheme using any arbitrary signal set of $2^n$ points and J dimensions. We denote the $j$th dimension of the signal point associated by the $i$-th symbol set by $m_{i,j}$ for $i = 0, 1, \ldots, 2^n - 1$ and $j = 1, 2, \ldots, J$. The J-dimensional modulation vector corresponding to a particular $n$-bit symbol is denoted by $i = (m_{i,1}, m_{i,2}, \ldots, m_{i,J})$. Table 5.1 shows the modulation vectors of coding for a few common modulation schemes.

**Table 5.1: Modulation Vectors representation of n-bit Symbols**

<table>
<thead>
<tr>
<th>S no</th>
<th>Modulation type</th>
<th>values</th>
<th>Modulation Vectors representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BPSK</td>
<td>$n$</td>
<td>$i = (s_1, s_2, \ldots)$</td>
</tr>
<tr>
<td>2</td>
<td>Orthogonaly</td>
<td>$2^n$</td>
<td>$s = (s_1, 0, \ldots, 0)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$1 = (s_1, 0, \ldots, 0)$</td>
</tr>
<tr>
<td>3</td>
<td>Bi-orthogonaly</td>
<td>$2^{n-1}$</td>
<td>$i = (s_1, 0, \ldots, 0)$, $(s_1, \ldots, 0)$</td>
</tr>
<tr>
<td>4</td>
<td>M-PSK</td>
<td>2</td>
<td>$1 = (i, s)$ \quad $= 2i / 2^n$</td>
</tr>
<tr>
<td>5</td>
<td>ASK</td>
<td>1</td>
<td>$i = d, 3d, 5d, \ldots$</td>
</tr>
</tbody>
</table>

**5.3.2 Decoder**

The Symbol-Based Turbo Code decoder diagram is similar to the traditional Turbo Code decoder. However, the constituent RSC decoders described in previous section operate on a single stage binary Recursive Systematic Convolutional trellis. In the Symbol- Based Turbo Code decoder, the constituent Recursive Systematic Convolutional decoders are modified to operate on $n$-bit symbols using the $n$-stage merged trellis diagram. We are introduced the following changes to the modified BCJR algorithm for accomplish this. The LLR values are defined differently to account for the $2^n$ possible symbol set values (it is instead of just two values, 0 and 1). The new LLR values are defined by

$$(d_k = i) = \log, i = 0, 1, 2, 3 \ldots, 2^n - 1$$

Where represented the received data streams of bits. This definition for the LLR values in coding allows for the easy conversion between symbol LLR values and symbol APP values in wireless communication. For an AWGN channel, the branch metrics are represented by:
\[ \text{Pr}\{ x_k | d_k, S_k, S_{k-1} \} = \text{Pr}\{ y_k | d_k, S_k, S_{k-1} \} = e^{1/N_0(y_k-b_d(d_k,S_{k-1},S_k))^2} \]

where \( b_s/p(d_k,S_{k-1}; S_k) \) is known as the modulator output, which is associated by the branch from state \( S_{k-1} \) to new state \( S_k \) at \( k \) if the corresponding input \( d_k \) is equal to \( i \). The \( b^{10}() \) values represented by the \( m_{i;j} \) values. Which is discussed earlier? These equations are generalized for any signal set by calculating them for each dimension and multiplying the resulting terms together in coding. This type of calculation is based on the assumption that each dimension of the modulation in coding vector is independent of all the others, and experience independent AWGN. Thirdly, the expression for the intrinsic information in coding [1, 3] is also modified to take into account of modulation scheme used in communication. A basic expression for the intrinsic information is given below.

\[
\log = 2x_{k,j}(m_{i,j},m_{0,j}) + (m_{20,j} - m_{i,j}),
\]

5.6

The value \( x_{k,j} \) is the \( j \)-th component of the \( k \)-th received systematic symbol in coding. Depending on the modulation scheme used in coding, the above expression for the communication may be simplified further. Lastly, the final hard-decision decoding in communication is performed by choosing the large LLR value and map the index of that LLR value back to an n-bit symbol in coding. For Recursive Systematic Convolutional encoders, whose trellis type representation does not have any parallel transitions in coding; the modified BCJR algorithm may be simplified. This is a practical case to consider in coding, because, as noted in previous section, parallel transitions of merged trellis diagram degrade the performance of the coding. In the modified type BCJR algorithm, summation is performed over previous states \( m_0 \) and the current states of \( m \). For a trellis diagram of coding with no parallel transitions a previous state \( m_0 \) and current state \( m \) indicate a unique transition in coding. Since a previous state of \( m_0 \) and an input uniquely define by current state \( m \) and an output \( Y \) in any trellis diagram, the branch metrics in equations can be redefined by terms of the indices \( m_0 \) and \( i \) as follows:

\[
\text{Pr}\{ k | s_{k-1} = m', d_k = i \} = )
\]

5.7

\[
\text{Pr}\{ k | s_{k-1} = m', d_k = i \} = (-)
\]

5.8
Furthermore, the forward and backward recursions will also change so that all the summations in coding are performed over the indices \( m_0 \) and \( i \). This type of simplification will reduce the complexity of the modified BCJR algorithm because all the summations in system is performed only over trellis branches which is actually exist. A further simplification can be made by modified BCJR algorithm by examining the values. The can be expressed in coding as follows by using Bayes rule and the simplification previously discussed:

\[
i (R_k, m' = Pr_k|s_{k-1} = m', d_k = i ) Pr\{s_{k-1} = m', d_k = i \} Pr\{d_k = i \}
\]

The first two terms are the branch metrics in coding for the systematic and parity data streams and are given by equations. The third term in equation is the APP of the input symbol \( d_k \).

Since the first two terms do not change between iterations in coding, they can be calculated once and merely multiplied by the most current APP values of \( Pr(d_k = i) \).

### 5.3.3 Orthogonal Modulation

The orthogonal modulation with Symbol Based Turbo Codes use a \( 2^n \) point orthogonal signal set to transmit the parallel data streams in turbo coding. In Table 5.2, the modulation vectors are represented by \( i = (0, \ldots, 0, i, 0, \ldots, 0) \), where \( Es \) is represented by transmitted energy per \( n \) bit symbol in equation. Therefore, \( Es = n E_b \), and \( Eb \) is represented by equivalent energy per bit. The modulation vectors \( i \) are zero in every position accept the \( i \) th position in equation, into which all the symbol energy of coding is placed. It is known that, the rows of a Hadamard matrix \( H_{2^n} \) can be used for the modulation vectors in equation. The generalized equation for the intrinsic information is given in equation 5.10. It can be simplified as follows.

\[
\text{Log} = (x_{k,i} - x_{k,0})
\]

The rates of these codes are represented by \( n/3 \) \((2n)\) where \( n \) is the symbol size in equation. Therefore, the code rates 1/6, 1/6, 1/8, 1/12, 1/19.2, 1/32 are represented by \( n = 1, 2, \ldots, 6 \) and this type of codes are suited to spread spectrum communication systems where the low code rates may be accommodated in the large spreading factor in communication system. In \( n = 6 \), the resulting codes have the same spectral efficiency as IS-95 up link code. It is composed by convolutional code with 1/3 code rate and 26-ary orthogonal signaling. A comparison of the bit error rate performance of these codes and a convolutional code in communication system with orthogonal signaling scheme with the same structure as the IS 95 up link code is shown
in section 5. The Symbol Based Turbo Codes in communication cannot be directly compared with IS-95 Up link because the IS 95 up link consists by non-coherent type reception in communication system while our study is main focused on the coherent reception scheme.

5.3.4 PSK Modulation scheme of M ary

in the second type of Symbol Based Turbo Code, we have studied M-ary PSK modulation scheme. The modulation vectors in equation is given by \( i = (\cos i, \sin i) \), where \( = 2\pi i/2^n \) and for larger symbol sizes, the i 's is grey coded to improve bit error rate performance. The generalized expression for the intrinsic type is given in equation 5.11 and it can be simplified as follows.

\[
\log \left( x_i (1 - \cos i - x_Q \sin i) \right) = -2
\]

Where \( x_i, x_Q \) are represented by the in phase and quadrature phase components of the k-th systematic modulation vector in equation. The rates of these codes are represented by \( n=6 \) and \( n \) is the symbol size. Hence, the code rates 1/6, 1/3, 1/2, 2/3, 5/6 are obtained at \( n = 1, 2, \ldots, 6 \). The performance of bit error rate of these codes are compared with convolutional codes in communication with a constraint length \( K \) equal to 9 with similar code rates in coding system.

5.3.5 Bi-orthogonal Type Modulation

In the third Symbol-Based Turbo Code we have studied as bi-orthogonal type of modulation. The modulation vectors are represented by \( = (0, \ldots, 0, 0, \ldots, 0) \). Therefore, the rows of a Hadamard matrix \( H_2^n \), the complement of a row, can be used for the modulation vectors in communication system.

The mapping between n-bit symbols and bi-orthogonal vectors is represented by Table 5.2.

Table 5.2: Mapping of n bit Symbols and Bi orthogonal Vectors

<table>
<thead>
<tr>
<th>Decimal point value</th>
<th>Binary</th>
<th>Modulation Vector in communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>N=1</td>
<td>N=2</td>
</tr>
<tr>
<td>0</td>
<td>000</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>0,+</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>0,+</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>0,+</td>
</tr>
</tbody>
</table>
If we express the n bit symbols in communication system in terms of the binary digits \(b_1, b_2, \ldots, b_n\), then the sign of the non-zero component in bi-orthogonal vector is given by \(b_1\) in equation. A 1 corresponds to a positive sign and a 0 to a negative sign representation. Furthermore, the position in which the symbol energy \((E_s)\) is placed, is represented by the expression \(b_2 b_3 \ldots b_n\) in equation if we set \(b_1 = 0\), and \(12 \ldots n\) if we set \(b_1 = 1\), where \(i\) represents the binary complement of bit \(b_i\) in equation. This mapping results of distance properties is given by \(d_{i,j} = 2E_s\) if \(i\) and \(d_{i,j} = 4E_s\). This mapping maximizes the bit error rate performance of the coding by separating symbols in communication whose binary values are complements of one another by the largest distance in communication system. The equation of the intrinsic information for this type of modulation scheme is calculated by equation 5.4 and with Table 5.2. There is no simplification of this equation because of the mapping required to maximize the bit error rate performance. The rates of these codes are represented by \(n/3 (2^n-1)\) where \(n\) is the symbol size in equation. Therefore the code rates 1/3, 1/3, 1/4, 1/6, 1/9, 1/16 are obtained for \(n = 1, 2, \ldots, 6\) in equation.

### 5.3.6 BPSK Modulation

In forth type of Symbol Based Turbo Code; we have studied as n-dimensional binary phase shift keying modulation. The modulation vectors are represented by equation \(i = (\ldots)\). The encoding system and modulation scheme of this code is same as traditional Turbo Code except to the interleaver design. The generalized equation for the intrinsic information can be simplified as follows,

\[
\log = 4 \cdot b_i
\]

Where the \(b_i\)'s is represented by binary digits of \(k\)-th input symbol in equation. The rate of this code is the same for any block size. We can also change the rate of the code in equation by adding an additional feed of forward generator and then puncturing the resulting encoded bit streams in communication system. Figure 5.4 shows an encoder of turbo code with feedback polynomial of \(1+D+D^3\), and feed-forward polynomials are represented by \(1+D+D^2+D^3\) and \(1+D^2+D^3\).
This Recursive Systematic Convolutional encoder can be used to obtain a code rates of $1/5$, $1/4$, $1/3$ $1/2$ in Turbo Code scheme by puncturing the parallel data streams. In Table 5.3, the puncturing patterns of code are shown. A 1 denotes a transmitted bit in communication system and 0 a non-transmitted or punctured bit. The puncturing patterns shown in Table 5.3 are used regardless of the symbol size in communication system and other type of uniform and non-uniform type puncturing patterns can also be used in communication system.

<table>
<thead>
<tr>
<th>Code rate</th>
<th>Bit stream in communication</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$X$</td>
</tr>
<tr>
<td>$1/5$</td>
<td>1111</td>
</tr>
<tr>
<td>$1/4$</td>
<td>1111</td>
</tr>
<tr>
<td>$1/3$</td>
<td>1111</td>
</tr>
<tr>
<td>$1/2$</td>
<td>1111</td>
</tr>
</tbody>
</table>

5.3.7 Parity Bits

The various parity code in communication before and after the Symbol Based Turbo Code encoder were also studied and first Symbol Based type Turbo Code with help of parity bits studied parses the input data into n-1 bit of sub block and adds a single parity bit to each sub block in communication to create n-bit symbols in equation. These symbols of communication
are then encoded by using the Symbol Based Turbo Code encoder. The interleaving will be performed by an n- bit basis. The rates of these codes are represented by \((n - 1)/3n\) for binary phase shift keying modulation. We have referred to these codes as Symbol Based Turbo Codes with full parity bit because the all parity bit is passed to the parallel data streams in communication. The trellis diagram of the RSC encoder is represented by \(2^{n-1}\) branches leaving each state. These branches correspond to all the possible n -1 bit in sub-blocks of communication. This type of trellis can also be obtained by using n-stage merged trellis diagram by removing every branch in coding scheme which does not have an even parity input symbol in communication system. The decoder must be uses this modified merged trellis diagram of constituent decoders. The second type of Symbol Based Turbo Code with parity bit are studied by only passes the parity m bit onto the encoded data streams in communication system. It is shown in Figure.

![Figure 5.5 Turbo Code of Symbol Based Encoder with Partial Parity](image)

**Figure 5.5 Turbo Code of Symbol Based Encoder with Partial Parity**

We call these codes as Symbol Based Turbo Codes with partial parity bit. This type of structure shows in a code rate of \((n-1)/(a+2j)\). When binary phase shift keying modulation is used in system, the code becomes \((n-1)/(3n-1)\) code. Hence code rates of 1/5, 1/4, 3/11 are obtained for \(n = 2, 3, 4\).
5.4 Simulation Setup

The bit error rate performance of the Symbol Based Turbo Codes is studied by simulation using an AWGN channel and a Rayleigh fading channel. The discrete type model of the additive white Gaussian noise channel is represented by the expression $k = k + k$, where $k$ represents the transmitted symbol and $k$ represents Gaussian random vector for independent and identically distributed (i.i.d.) components with mean zero and variance $N_0/2$. Since, the discrete model in communication for Rayleigh fading channel is represented by the expression $k = a_k + k$, where $k$ and $k$ are the same as mentioned above, and the $a_k$'s are independent and identically distributed random variables with a Rayleigh distribution of the form $f(a) = 2ae^{-a^2}$ for $a \geq 0$. In Rayleigh fading channel, the decoder has the $a_k$'s into the decoding algorithm. In other words for decoding, it is assumed that the receiver in communication can determine by multiplicative fading factors. The discrete type model for both channels is shown in Figure.

Most of the simulations of Symbol-Based Turbo Code encoder were performed by Figure 5.6. The Recursive Systematic Convolutional encoders have $M = 4$ memory elements and generator polynomial of $(37, 21)_8$. Unless otherwise noted, 20 iterations in wireless communication are performed by the decoder in circuit. The data rate is available 9.6 Kbits/sec, the IS-95 delay specification of 20 ms available to a maximum block length of $N = 192$ bits. This type of block length is used for most of the simulations. However, other block lengths were also simulated for comparison in communication. As matter discussed previously, when the block size in communication, $n$, exceeds the number of memory elements, parallel transitions occur in the merged trellis diagram which severely degrade the performance of the turbo code. For symbol sizes used in communication 5 and 6, RC type encoders with the value of $M = n$ are used in communication. Some of the Symbol Based Turbo Codes in communication were also simulated using the Recursive Systematic Convolutional encoder shown in Figure, in which $M = 3$ memory elements and two feed forward outputs in coding. In each case, the block size and the code rate is noted. The Symbol-Based Turbo Codes for bi-orthogonal, orthogonal, M-ary PSK, and BPSK modulation were simulated. For binary phase shift keying modulation, the Symbol-Based Turbo Code is simulated with no parity bit, full parity bit and partial parity bit.
\[ y_k = x_k a_k + z_k \]

\[ y_k = x_k + z_k \]

**AWGN Channel model**

**Rayleigh Fading Channel model**

**Figure 5.6: Rayleigh Fading, AWGN Channel Models**

### 5.5 Numerical Results

#### 5.5.1 Symbol-Based Turbo Codes BER Performance

##### 5.5.1.1 Orthogonal Modulation

Figure shows the bit error rate performance of Symbol-Based Turbo Codes with orthogonal modulation over an Rayleigh fading channel and AWGN channel for symbol sizes \( n = 1, 2, \ldots, 6 \). The rates of this code are \( n/3(2^n) \). For a symbol size of \( n/6 \), the resulting code will have the same spectral efficiency as the IS-95 Up-link code in communication; it is composed by rate 1/3 convolution code with 26-ary orthogonal signaling. In Figure, the bit error rate performance of the Symbol-Based Turbo Codes are compared with a convolutional code in communication, constraint length \( K = 9 \), with 26-ary orthogonal signaling. The Symbol-Based Turbo Codes cannot be directly compared in communication system with the IS-95 Up-link because the IS-95 Up-link has non-coherent reception while our study has been focused on coherent reception in communication. The Figure show that the Symbol-Based Turbo Code with orthogonal modulation scheme with symbol size of \( n/6 \) performs about 1.4 dB better than the convolutional code with orthogonal signaling at a bit error rate of \( 10^{-3} \) for an AWGN channel and it is about 2.9 dB better for a Rayleigh fading channel scheme. It is clear that that the Symbol-Based Turbo Code with non-coherent reception scheme will be perform better than the IS-95 Up-link code.

##### 5.5.1.2 M-ary Phase Shift Keying Modulation

The M-ary PSK modulation with Symbol Based Turbo Codes has rates of \( n/6 \), where \( n \) represents the symbol size. The bit error rate performance of these codes in communication system are compared with rate 1/3 and rate 1/2 convolutional codes with constraint length \( K = 93 \).
9 for both Rayleigh fading channels and AWGN channel. For a bit error rate of 10^{-3}, the Symbol-Based Turbo Codes with M-ary Phase shift keying modulation performs about 0.7 dB better than convolutional codes of the same rate over an AWGN channel, and about 2.2 dB better over a Rayleigh fading channel in communication system.

5.5.1.3 Bi-orthogonal Modulation scheme

The third type of Symbol-Based Turbo Code simulated and uses bi orthogonal modulation scheme. The rates of these codes are represented by n/3(2^{n-1}). The Symbol-Based Turbo Code with bi-orthogonal modulation scheme obtains a bit error rate performance of 10^{-3} at an signal to noise ratio of 0.68 dB with symbol size of n / 4. For a symbol size of n = 6, a bit error rate performance of 10^{-3} is obtained at signal to noise ratio of 0.30 dB. With comparison, the Symbol-Based Turbo Code of bi orthogonal modulation scheme for a symbol size n = 3 with rate 1/8 were also simulated for other block lengths N in communication system. A bit error rate performance of 10^{-3} is obtained at signal to noise ratio of 1.27 dB for N = 96 in wireless communication, at 0.93 dB for N = 192, at 0.45 dB for N = 512, and at 0.22 dB for N = 1024 in wireless communication scheme.

5.5.1.4 BPSK Modulation

The Symbol Based Turbo Codes with n-dimensional binary phase shift keying modulation is simulated. The bit error rate performance of these codes improve only less than 0.1 dB, with increasing the symbol size from n = 1 to n = 2. For value of n = 3, 4….. The bit error rate performance of this code is essentially the same as the n = 2 code. However, the strength of these codes can be evaluated when we consider the 5 and 10 iteration bit error rate performance with symbol sizes n = 1, 2.

It is Symbol-Based Turbo Code 0f M = 4, R = 1/3, with binary phase shift keying modulation for 5 and 10 iterations with block sizes N = 192 and 512. For both type of block sizes, the Symbol Based Turbo Code with symbol size of n = 2 requires about 5 iterations to obtain the same bit error rate performance like traditional Turbo Code with n = 1 and no of iteration equal to 10.

The same type of trend is also observed with M = 4, R = 1/4 Symbol Based Turbo Code with binary phase shift keying modulation with block length of N = 192. However, for a block length of N = 512 with 8 iterations of the n = 2 code are required to obtain the same bit error rate performance as the n = 1 code with 10 iterations in coding scheme. Therefore, it is clear
that, for large block sizes in coding scheme, that Recursive Systematic Convolutional encoders with only one feed-forward output be used in this type of coding scheme. There is less than 20 iterations are performed in coding. It is shows the degradation in bit error rate performance of Symbol-Based Turbo Codes with binary phase shift keying modulation. It is clear that as the symbol size in coding increases, the degradation in bit error rate performance decreases and we conclude that the modified BCJR algorithm in coding converges faster as the symbol size is increased in system. The same type of trends is also observed when bi-orthogonal or orthogonal modulations are used in system. Table 5.4 shows a comparison of the degradation in bit error rate performance for various modulation techniques with symbol size of $n=4$.

**Table 5.4: Bit error rate Performance Degradation of Symbol-Based Turbo Codes for Various Modulation Techniques as Compared with Symbol Size $n=4$, 20 Iterations**

<table>
<thead>
<tr>
<th>Modulation type</th>
<th>No of Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>iteration=5</td>
</tr>
<tr>
<td>Orthogonal</td>
<td>0.0801</td>
</tr>
<tr>
<td>Bi-orthogonal</td>
<td>0.0710</td>
</tr>
<tr>
<td>BPSK</td>
<td>0.0900</td>
</tr>
</tbody>
</table>

**5.5.1.5 Parity Bits of turbo code**

Full parity with Symbol-Based Turbo Codes was the first class of codes in communication system. We have studied the combine Symbol-Based Turbo Codes with parity bit codes in communication. These codes, as discussed in previous section, parse the input data bit into $n-1$ bit sub blocks and maps them to $n$-bit symbols in communication system by adding a single parity bit. These symbols are then passed through a Symbol Based Turbo Code encoder in wireless communication. The rates of these code is $(n-1)/3n-1$ for binary phase shift keying modulation. The second set of codes studied is known as Symbol-Based Turbo Code with partial parity bit. In this type of codes, the parity bit added to the $n-1$ bit sub blocks are passed to the encoded data bit and not to the systematic type data bit. The rates of these codes are represented by $(n -1)/(3n - 1)$ for binary phase shift keying modulation. The BER performance of these codes is tabulated. For both parity schemes of this code, the $n=2$ code is essentially a repetition code in communication system. These codes perform worse type
code than the traditional Turbo Code in communication with no parity bit. For $n = 3$ and 4, both parity codes perform the 0.2 dB better than the traditional Turbo Code in communication system.

BER Performance of Symbol-Based Turbo Codes with $M = 4$, $N = 192$, BPSK Modulation and Full Parity bits are performed. However, for $R = 1/4$ code rate, the full parity code requires a symbol size of $n = 4$ and the partial parity code requires a symbol size of $n = 3$. Since the decoder complexity is a function of the symbol size $n$ in coding scheme, as discussed in previous section, the partial parity bit code may obtain the same bit error rate performance as the full parity bit code with lower decoder complexity in system.

### 5.5.2 Complexity and Memory

With merging of $n$ sections of the full parity trellis diagram which results in a reduction in the effective block length by a factor of $1/n$ in coding. This shorter effective block length in coding means that fewer and from the forward and backward recursions in the BCJR algorithm need to be stored in memory. Similarly, the shorter effective block length of coding means that the modified BCJR algorithm in system is calculated over fewer stages in communication, thus resulting in a reduction with number of computations. On the other hand, the merged trellis diagram in communication has more branches leaving each state. It is specifically $2^n$ branches. These extra branches also increases the number of calculations required per stage in coding. Therefore, more branches translate into more LLR values that need to be stored in communication system. Finally, the use of a multi-dimensional type signal space results in more complex branch metric calculations in communication and it requires more memory storage in coding. The computational complexity and memory requirements of the Symbol-Based Turbo Code decoder relative to the traditional Turbo Code model with $n = 1$, are shown in Table.

**Table 5.5: Per Iteration Computational Complexity and Relative Memory Requirements of the Symbol-Based Turbo Code Decoder or RAM requirements**

<table>
<thead>
<tr>
<th>Memory Size n</th>
<th>Symbol Size n</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory of decoder</td>
<td></td>
<td>3</td>
<td>1</td>
<td>0.79</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>1</td>
<td>0.68</td>
<td>0.68</td>
</tr>
<tr>
<td>Complexity of decoder</td>
<td></td>
<td>3</td>
<td>1</td>
<td>1.0</td>
<td>1.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>1</td>
<td>1.00</td>
<td>1.30</td>
</tr>
</tbody>
</table>
RAM requirements of Symbol-Based Turbo Code decoder are less than the traditional Turbo Code. Furthermore, the Read Only Memory (ROM) used to store the interleaver structure is also reduced by a factor of $1/n$. The relative computational complexity in communication is given by the expression $2^{n-1}/n$. This result is intuitively satisfying in equation because a merged trellis diagram has $2^n$ branches leaving each state in communication. It is consist by $n$ input bits. The traditional Turbo Code, $n = 1$, has 2 branches leaving each state of the Recursive Systematic Convolutional trellis, and one stage of this trellis corresponds to a single input bit representation. Therefore $(2^n/n)/(2/1) = 2^n-1/n$. Therefore, the computational complexity per iteration in symbol sizes of $n = 1, 2$ are approximately the same. The Symbol-Based Turbo Codes with binary phase shift keying modulation, symbol size $n = 2$ require approximately 5 iterations to obtain the same bit error rate performance obtain in the traditional Turbo Code, $n = 1$, with 10 iterations. Overall, the Symbol-Based Turbo Codes with $n = 2$ result in an improved performance in communication system for the same number of decoding iterations (it is equivalently a reduction for number of iterations with same performance), as well as a reduction in the required memory size in communication system with respect to the traditional Turbo Code, $n = 1$, with no extra cost. For larger symbol sizes in communication $n = 3, 4$, the increase in computational complexity requirement is partially compensated by a reduction in the number of iterations required in communication system.

We compare the computational complexity in communication and memory requirements of the Symbol-Based Turbo Code decoder by using Viterbi decoding of a convolutional code with constraint length $K = 9$. The Symbol Based Turbo Code decoder type with 5 iterations in communication require more computations than the convolutional coding scheme, specifically, 3.7 to 7.4 times more, depending on the symbol size in communication, $n$, and the number of memory elements, $M$, in the encoders.

Table 5.6: Total Number of Decoding Operations and Words of RAM Memory Required for the Symbol-Based Turbo Codes (5 iterations) and a Convolutional Code ($K = 9$)

<table>
<thead>
<tr>
<th>Computations</th>
<th>Memory Size . (M)</th>
<th>Memory Size (n)</th>
<th>Conv. Code (K = 9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Size</td>
<td>3</td>
<td>2870</td>
<td>2845</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>21</td>
<td>256</td>
</tr>
</tbody>
</table>
Therefore, the Symbol-Based Turbo Code decoder requires much less random access memory storage capacity than the convolutional code in communication system.

- Trade-offs between code rate, BER performance, decoder complexity and spectral efficiency. In Upper Bound for MAP Decoder, BER performance will be decrease from $10^{-1}$ to $10^{-7}$ as $E_b/No$ (dB) increase from 0 dB to 8 dB. My simulation result is very close to union bound result.

- Performance of 2-state RSC is lower as compared to NRC codes. The coding gain of Non recursive code is 0.5 dB above than to recursive code.

- Code with FER simulation of large block size and Uniform interleaver, BER performance will be decrease from $10^0$ to $10^{-2}$ as $E_b/No$ (dB) increase from -0.5 dB to 1.5 dB. The simulation result of uniform terminated code is better than unterminated code.

- With increasing the no of iteration, the bit error rate performance will be improved.

Turbo code Bit Error Rate and Frame Error Rate simulation with large block size

- Uniform interleaver type
  BER performance will be decrease from $10^{-1}$ to $10^{-5}$ as $E_b/No$ (dB) increase from -0.5 dB to 1.5 dB. The simulation result of uniform terminated code is better than unterminated code.
  FER performance will be decrease from $10^0$ to $10^{-2}$ as $E_b/No$ (dB) increase from -0.5 dB to 1.5 dB. The simulation result of uniform terminated code is better than unterminated code.

- Regular interleavers type
  FER performance of Uniform Interleaver is better than helical, square and rectangular interleaver.

- Randomized interleavers type
  BER performance of uniform interleaver is poor than square interleaver.
  FER performance of uniform interleaver is poor than square interleaver

Turbo code Bit Error Rate and Frame Error Rate simulation with small block size
Unpunctured rate of 1/3 symmetric Turbo codes with help of encoder memory $\nu = 2$ and generator $(1; 5/7)$. It is clear that we have chosen a very small block size $\tau = 64$ to analyze the effect of termination in turbo coding where it should be most pronounced scheme.

The trellis termination improves the performance of Turbo coding so codes at high SNR with encoder memory of $\nu = 2$. It is scene that, the frame error rate (FER) performance is better in this type of coding and the bit error rate (BER) performance is also improved in turbo coding.

**Uniform interleaver**

BER and FER performance of uniform JPL terminated interleaver is improved at high $E_b/No$ with respect of unterminated type

Improvement due to coding

If $E_b/No$ is given 14 db, coding give the better performance. If $E_b/No = 10$ db then coding give no error performance. There is degradation.

In BPSK, code $(15, 11)$, Block length =15 bit, $P_r/No = 43776$, $R = 4800$ bit/S

Improvement Due to Coding = $112 \times 10^{-4}/1.94 \times 10^{-6} = 58$ times

### 6.2 Interleaver Parameter Design

We have proposed a new interleaver parameter design technique, and shown that it performs better than other regular and partly-randomized designs. For the case of small interleavers design, it is shown that correct termination improves the performance for an average interleaver; its effect on Turbo codes with optimized interleavers is negligible in coding scheme. It is considering that designing an interleaver without termination is simpler and that there is also a minor gain in code rate. The design of non-terminating interleavers should be preferred for all block sizes in coding scheme. Using our simulated annealing design technique it is easier to include restrictions which make the interleaver correctly-terminating or odd-even. While the semi-random algorithm serves well for specifying interleaver spread.

We analyzed that our algorithm in coding is better suited for more sophisticated design criteria in turbo code. By utilizing some performance enhancement techniques, the complexity of the energy function grows only as $O(\tau)$, making it suitable for use with large block sizes. However, increasing the block size also requires the simulated annealing algorithm to perform more perturbations at every temperature, so that the overall complexity of the design technique actually grows larger than $O(\tau)$. 
Turbo Coding is an effective method for combining Turbo Codes with different modulation schemes.

Low-rate Symbol-Based Turbo Codes which use bi-orthogonal or orthogonal modulation display an improvement in bit error rate performance as the symbol size is increased in communication. Furthermore, these types of codes perform better than constraint length 9 convolutional codes with the same modulation scheme, specifically, 2.9 dB better for a Rayleigh fading and 1.4 dB better for an AWGN channel for the orthogonal modulation case.

The high-rate Symbol-Based Turbo Codes of M-ary PSK modulation perform 0.7 dB better than convolutional codes of the same rate and similar decoding complexity over an AWGN channel and it is about 2.2 dB better over a Rayleigh fading channel.

The bit error rate performance of the Symbol-Based Turbo Codes analyzed only degrades 1.25 to 1.5 dB between an AWGN channel and Rayleigh fading channel. In comparison, the convolutional codes displayed 3 to 4 dB degradation.