EFFICIENT COMMUNICATION SYSTEM USING CONVOLUTION ENCODER

Submitted by

ABHIROOP BHATTACHARYA
NANDAN BHOWMICK
PRANABESH BHATTACHARYA
KAUSTAB MADHAB CHATTERJEE
SAYANTAN DUTTA

Under Guidance of : Mr. Surojit Kundu
Department of ECE
Institute of Science & Technology
CONTENTS

CHAPTERS

CHAPTER-1 INTRODUCTION

1.1 OBJECTIVE 2
1.2 FOCUS OF THE THESIS 2
1.3 ORGANIZATION OF THE THESIS 3

CHAPTER-2 OVERVIEW

2.1 BASIC COMMUNICATION SYSTEM 4
2.2 INTERLEAVING 5
2.3 FORWARD ERROR CORRECTION 7
2.4 FEC ARCHITECTURE 7
2.5 CONSTELLATION DIAGRAM 8

CHAPTER-3 EVOLUTION OF CODES

3.1 BLOCK CODE 9
3.2 CYCLIC CODES 11
3.3 TURBO CODES 12
3.4 CONVOLUTIONAL CODE 13
3.5 REFERENCE 13
3.6 APPLICATION OF TURBO CODE 13
CHAPTER-7 DECLARATION

7.1 CONCLUSION 32
7.2 FUTURE WORKS 32
7.3 REFERENCES 33
<table>
<thead>
<tr>
<th><strong>LIST OF FIGURES</strong></th>
<th><strong>PAGE NO.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>FIG-1.1 BASIC COMMUNICATION SYSTEMS</td>
<td>4</td>
</tr>
<tr>
<td>Fig-2.4.1 FEC ARCHITECTURE</td>
<td>8</td>
</tr>
<tr>
<td>FIG-2.5.1 CONSTELLATION DIAGRAM</td>
<td>9</td>
</tr>
<tr>
<td>FIG-4.1.1 CONVOLUTION ENCODING</td>
<td>15</td>
</tr>
<tr>
<td>FIG 4.2.1 STATE DIAGRAM</td>
<td>16</td>
</tr>
<tr>
<td>FIG. 4.3.1 TREE DIAGRAM</td>
<td>18</td>
</tr>
<tr>
<td>FIG. 4.4.1 TRELLIS DIAGRAM</td>
<td>19</td>
</tr>
<tr>
<td>FIG. 4.6.1 BASIC CONVOLUTIONAL ENCODER (2,1,2)</td>
<td>21</td>
</tr>
<tr>
<td>FIG-4.6.2 BASIC CONVOLUTIONAL ENCODER(2,1,2)</td>
<td>22</td>
</tr>
<tr>
<td>FIG-4.6.2 BASIC CONVOLUTIONAL ENCODER(3,1,2)</td>
<td>23</td>
</tr>
<tr>
<td>FIG-4.7.1 DECODING PROCEDURE</td>
<td>23</td>
</tr>
<tr>
<td>FIG-4.7.2 BER PERFORMANCE OF FOFT AND HARD DECODING</td>
<td>24</td>
</tr>
<tr>
<td>FIG-5.1.1 MATLAB ENCODER MODEL</td>
<td>25</td>
</tr>
<tr>
<td>FIG-5.3.1 A PERFECT CONVOLUTIONAL ARCHITECTURE</td>
<td>28</td>
</tr>
<tr>
<td>FIG 5.5.1 DECODING ARCHITECTURE WITH PUNCTURING</td>
<td>30</td>
</tr>
<tr>
<td>FIG-5.5.2 PUNCTURE OUTPUT</td>
<td>30</td>
</tr>
</tbody>
</table>
LIST OF TABLES

FIG-2.1 INTERLEAVING 6
FIG 4.2.1 STATE DIAGRAM 16
Fig-5.4.1 PUNCTURED COVOLUTIONAL CODES 29
[1] INTRODUCTION

1.1 OBJECTIVE

Nowadays highly reliable data transmission is hugely needed. There are several types of coding procedures. Out of these convolution coding is mostly used because of its higher ability to detect and correct errors more than the others. Hereby it is specially notified that how to encode data using convolutional encoding method and how to minimize the error using convolution coding and also simulating and designing convolutional encoder using matlab for observing and minimizing bit error rate.

1.2 FOCUS OF THE THESIS

The probability of error can be reduced by transmitting more bits than needed to represent the information being sent, and convolving each bit with neighbouring bits so that if one transmitted bit got corrupted, enough information is carried by the neighbouring bits to estimate what the corrupted bit was. This approach of transforming a number of information bits into a larger number of transmitted bits is called channel coding, and the particular approach of convolving the bits to distribute the information is referred to as convolution coding. The ratio of information bits to transmitted bits is the code rate (less than 1) and the number of information bits over which the convolution takes place is the constraint length. The main focus of this project is to design a convolutional encoder so as to observe
and minimize the errors and also observe and minimize the Bit error rate through simulation and design of convolution encoder using matlab. The output received for a given input can fulfill our desire of having an error free transmission.

1.3 ORGANISATION OF THE THESIS

Convolutional coding has a vast application in the industrial field. Convolutional codes are used extensively in numerous applications in order to achieve reliable data transfer, including digital video, radio, mobile communication, and satellite communication. To design encoder MATLAB is preferred for its easy accessibility and better bit error rate calculation capability. Hardware implementation of the convolutional coding is hugely expensive. It is mostly done in the industry. It is out of scope of this project. Here we have implemented the convolution coding with MATLAB.
**[2] OVERVIEW**

### 2.1 BASIC COMMUNICATION SYSTEMS

**FIG-1.1**

SOURCE-Information input source. Generates the required information for transmission.

SOURCE CODER-It is used for sampling and quantizing the given input. It consists of A/D converter, sampler, quantizer.

CHANNEL CODER-redundant bit is added to the given input so as to check the output whether it is error free or not.

MODULATOR- modulation is the process of conveying a message signal, for example a digital bit stream or an analog audio signal, inside another signal that
can be physically transmitted. A device that performs modulation is known as a modulator.

CHANNEL-The path through which data is transmitted is channel. It can be wired or wireless.

DEMODULATOR- Demodulation is the act of extracting the original information-bearing signal from a modulated carrier wave. A demodulator is an electronic circuit that is used to recover the information content from the modulated carrier wave.

CHANNEL DECODER- A decoder is a device which does the reverse of an encoder, undoing the encoding so that the original information can be retrieved.

SOURCE DECODER- A decoder does the opposite of the source encoder

2.2 INTERLEAVING

Interleaving is frequently used in digital communication and storage systems to improve the performance of forward error correcting codes. Many communication channels are not memoryless: errors typically occur in bursts rather than independently. If the number of errors within a code word exceeds the error-correcting code's capability, it fails to recover the original code word. Interleaving ameliorates this problem by shuffling source symbols across several code words, thereby creating a more uniform distribution of errors.

Interleaver designs include:
• rectangular (or uniform) interleavers (similar to the method using skip factors described above)
• convolutional interleavers
• random interleavers (where the interleaver is a known random permutation)
• S-random interleaver (where the interleaver is a known random permutation with the constraint that no input symbols within distance S appear within a distance of S in the output).
• Another possible construction is a contention-free quadratic permutation polynomial (QPP). It is used for example in the 3GPP Long Term Evolution mobile telecommunication standard.

Example:

![Diagram](FIG-2.1)

Transmission without interleaving:

<table>
<thead>
<tr>
<th>Order Bits Received</th>
<th>29</th>
<th>10</th>
<th>30</th>
<th>30</th>
<th>34</th>
<th>35</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth Of Interleaving</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>11</td>
<td>3</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>Length</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error-free message: aaaabbbccccdddeeefeffffgigg
Transmission with a burst error: aaaabbbccc____deeefeffffgigg
The codeword cddd is altered in four bits, so either it cannot be decoded at all or it might be decoded incorrectly.

With interleaving:

<table>
<thead>
<tr>
<th>Error-free code words:</th>
<th>aaaabbbccccdddeeeeffffgggg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interleaved:</td>
<td>abcdefabcdefabcdefabcdefg</td>
</tr>
<tr>
<td>Transmission with a burst error:</td>
<td>abcdefgabcd___bedfabcdefg</td>
</tr>
</tbody>
</table>

### 2.3 FORWARD ERROR CORRECTION

In telecommunication and information theory, **forward error correction (FEC)** (also called **channel coding**) is a system of error control for data transmission, whereby the sender adds systematically generated redundant data to its messages, also known as an **error-correcting code (ECC)**. FEC is accomplished by adding redundancy to the transmitted information using a predetermined algorithm. A redundant bit may be a complex function of many original information bits. The original information may or may not appear literally in the encoded output; codes that include the unmodified input in the output are **systematic**, while those that do not are **non-systematic**.
2.4 FEC ARCHITECTURE

Fig-2.4.1
2.5 CONSTELLATION DIAGRAM

A **constellation diagram** is a representation of a signal modulated by a digital modulation scheme such as quadrature amplitude modulation or phase-shift keying. It displays the signal as a two-dimensional scatter diagram in the complex plane at symbol sampling instants. In a more abstract sense, it represents the possible symbols that may be selected by a given modulation scheme as points in the complex plane. Measured constellation diagrams can be used to recognize the type of interference and distortion in a signal.

![Fig-2.5.1](image-url)
[3] EVOLUTION OF CODES

3.1 BLOCK CODE

A block code transforms a message $m$ consisting of a sequence of information symbols over an alphabet $\Sigma$ into a fixed-length sequence $c$ of $n$ encoding symbols, called a code word. In a linear block code, each input message has a fixed length of $k < n$ input symbols. The redundancy added to a message by transforming it into a larger code word enables a receiver to detect and correct errors in a transmitted code word, and – using a suitable decoding algorithm – to recover the original message. The redundancy is described in terms of its information rate, or more simply – for a linear block code – in terms of its code rate, $k/n$.

**EXAMPLE**

- **message**: $k$-tuple $u=(u_1,u_2,\ldots,u_k)$

- **code word**: $n$-tuple $v=(v_1,v_2,\ldots,v_n)$
Some kinds of block codes

- **Linear Block Codes:**
  
  - Sum of any 2 code words results in a third unique codeword.

- **Systematic Code:**
  
  - The data bits also are present in the generated codeword.

- **BCH:**
  
  - Generalization of Hamming code for multiple error correction.
  
  - Very special class of linear codes known as Goppa codes.

- **Cyclic Codes:**
  
  - Important subclass of linear block codes where encoding and decoding can be implemented easily.
  
  - Cyclic shift of a code word yields another code word.

- **Group Codes:**
  
  - Same as linear block codes. Also known as generalized parity check codes.
3.2 CYCLIC CODES

In coding theory, cyclic codes are linear block error-correcting codes that have convenient algebraic structures for efficient error detection and correction. In shortened codes information symbols are deleted to obtain a desired blocklength smaller than the design blocklength. The missing information symbols are usually imagined to be at the beginning of the codeword and are considered to be 0. Therefore, \( n-k \) is fixed, and then \( k \) is decreased which eventually decreases \( n \). Note that it is not necessary to delete the starting symbols. Depending on the application sometimes consecutive positions are considered as 0 and are deleted.

All the symbols which are dropped need not be transmitted and at the receiving end can be reinserted. To convert \((n,k)\) cyclic code to \((n-b,k-b)\) shortened code, set \( b \) symbols to zero and drop them from each codeword. Any cyclic code can be converted to quasi-cyclic codes by dropping every \( b \)th symbol where \( b \) is a factor of \( n \). If the dropped symbols are not check symbols then this cyclic code is also a shortened code.

Cyclic codes can be used to correct errors, like Hamming codes as a cyclic codes can be used for correcting single error. Likewise, they are also used to correct double errors and burst errors.

EXAMPLE:

If the given cyclic code length \( n \).

Indeed, all we need to do is to find all factors of
Problem: Find all binary cyclic codes of length 3.

Solution: Since
\[ x^3 - 1 = (x + 1)(x^2 + x + 1) \]
both factors are irreducible in \( GF(2) \)

we have the following generator polynomials and codes.

<table>
<thead>
<tr>
<th>Generator polynomials</th>
<th>Code in ( R_3 )</th>
<th>Code in ( V(3,2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( R_3 )</td>
<td>( V(3,2) )</td>
</tr>
<tr>
<td>( x + 1 )</td>
<td>( {0, 1 + x, x + x^2, 1 + x^2} )</td>
<td>( {000, 110, 011, 101} )</td>
</tr>
<tr>
<td>( x^2 + x + 1 )</td>
<td>( {0, 1 + x + x^2} )</td>
<td>( {000, 111} )</td>
</tr>
<tr>
<td>( x^3 - 1 ) (= 0)</td>
<td>( {0} )</td>
<td>( {000} )</td>
</tr>
</tbody>
</table>

3.3 TURBO CODES

Shannon’s channel coding theorem guarantees the existence of codes that can achieve arbitrary small probability of error if the data transmission rate is smaller than the channel capacity. Simple Viterbi-decoded convolutional codes are now giving way to turbo codes, a new class of iterated short convolutional codes that closely approach the theoretical limits imposed by Shannon's theorem with much less decoding complexity than the Viterbi algorithm on the long convolutional codes that would be required for the same performance. Concatenation with an
outer algebraic code (e.g., Reed-Solomon) addresses the issue of error floors inherent to turbo code designs.

### 3.4 CONVOLUTIONAL CODE

Convolutional coding is a forward error correcting code. Generates $n$-bit message block from a $k$-bit block. Different from block codes in that encoder has memory order $m$ or constraint length. Must be implemented using sequential logic circuit. Decoders have high complexity (decoding operations) per output bit. More suited for low-speed digitized voice traffic (lower integrity) than for high-speed data needing high integrity. If the decoder loses or makes a mistake, errors will propagate. Performance depends on the constraint length.

### 3.5 DIFFERENCES

<table>
<thead>
<tr>
<th>Convolutional Codes</th>
<th>Turbo Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursive</td>
<td>Non-recursive</td>
</tr>
<tr>
<td>Non-systematic</td>
<td>Systematic</td>
</tr>
<tr>
<td>Without Interleaver</td>
<td>Use Interleaver</td>
</tr>
</tbody>
</table>

### 3.6 APPLICATIONS OF TURBO CODES

Turbo code is currently adopted as the channel coding schemes in many next-generation communication systems

- WCDMA, CDMA2000
CONVOLUTIONAL CODING

4.1 CONVOLUTION ENCODING

To convolutionally encode data, start with \( k \) memory registers, each holding 1 input bit. Unless otherwise specified, all memory registers start with a value of 0. The encoder has \( n \) modulo-2 adders (a modulo 2 adder can be implemented with a single Boolean XOR gate, where the logic is: 0+0 = 0, 0+1 = 1, 1+0 = 1, 1+1 = 0), and \( n \) generator polynomials — one for each adder (see figure below). An input bit \( m_1 \) is fed into the leftmost register. Using the generator polynomials and the existing values in the remaining registers, the encoder outputs \( n \) bits. Now bit shift all register values to the right (\( m_1 \) moves to \( m_0 \), \( m_0 \) moves to \( m_{-1} \)) and wait for the next input bit. If there are no remaining input bits, the encoder continues output until all registers have returned to the zero state. The figure below is a rate 1/3 \((m/n)\) encoder with constraint length \((k)\) of 3. Generator polynomials are \( G_1 = (1,1,1) \), \( G_2 = (0,1,1) \), and \( G_3 = (1,0,1) \). Therefore, output bits are calculated (modulo 2) as follows:

\[
\begin{align*}
    n_1 &= m_1 + m_0 + m_{-1} \\
    n_2 &= m_0 + m_{-1} \\
    n_3 &= m_1 + m_{-1}.
\end{align*}
\]
Rate 1/3 non-recursive, non-systematic convolutional encoder with constraint length 3

**STATE DIAGRAM**

The output depends on the current input and the state of the encoder (i.e. the contents of the shift register). Here the dotted line denotes 0 input & solid line denotes 1.

1. A convolutional encoder can be treated as a finite state machine.
2. The contents of the shift registers represent the states. The output of a code block vt at time t depends on the current state ¾t and the information block ut.
3. Each change of state ¾t → ¾t+1 is associated with the input of an information block and the output of a code block.
4. The state diagram is obtained by drawing a graph. In this graph, nodes are possible states and the state transitions are labeled with the appropriate inputs and outputs (ut/vt). In this course we only consider the convolutional encoder with state diagrams that do not have parallel transitions.

![State Diagram](image)

<table>
<thead>
<tr>
<th>state</th>
<th>Binary description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>00</td>
</tr>
<tr>
<td>b</td>
<td>10</td>
</tr>
<tr>
<td>c</td>
<td>01</td>
</tr>
<tr>
<td>d</td>
<td>11</td>
</tr>
</tbody>
</table>

FIG 4.2.1
4.3 TREE DIAGRAM

The tree diagram attempts to show the passage of time as we go deeper into the tree branches. It is somewhat better than a state diagram but still not the preferred approach for representing convolutional codes. Here instead of jumping from one state to another, we go down branches of the tree depending on whether a 1 or 0 is received.

The first branch indicates the arrival of a 0 or a 1 bit. The starting state is assumed to be 000. If a 0 is received, we go up and if a 1 is received, then we go downwards. In the figure, the solid lines show the arrival of a 0 bit and the shaded lines the arrival of a 1 bit. The first 2 bits show the output bits and the number inside the parentheses is the output state.

1. A code tree of a binary (n, k,m) convolutional code presents every codeword as a path on a tree.
2. For input sequences of length L bits, the code tree consists of (L + m + 1) levels. The single leftmost node at level 0 is called the origin node.
3. At the first L levels, there are exactly 2k branches leaving each node. For those nodes located at levels L through (L + m), only one branch remains. The 2kL rightmost nodes at level (L + m) are called the terminal nodes.
4. As expected, a path from the single origin node to a terminal node represents a codeword; therefore, it is named the code path corresponding to the codeword.
4.4 TRELLIS DIAGRAM

1. A code trellis as termed by Forney is a structure obtained from a code tree by merging those nodes in the same state.
2. Recall that the state associated with a node is determined by the associated shift-register contents.
3. For a binary \((n, k, m)\) convolutional code, the number of states at levels \(m\) through \(L\) is \(2^K\), where \(K = \sum_{j=1}^{k} K_j\) and \(K_j\) is the length of the \(j\)th shift register in the encoder; hence, there are \(2^K\) nodes on these levels.

4. Due to node merging, only one terminal node remains in a trellis.

5. Analogous to a code tree, a path from the single origin node to the single terminal node in a trellis also mirrors a codeword.

4.5 IMPULSE RESPONSE, TRANSFER FUNCTION & CONTRAINT LENGTH

A convolutional encoder is called so because it performs a convolution of the input stream with the encoder's impulse responses:

\[
y_i^j = \sum_{k=0}^{\infty} h_k^j x_{i-k},
\]

where \(x\) is an input sequence, \(y_i^j\) is a sequence from output \(j\) and \(h^j\) is an impulse response for output \(j\).
A convolutional encoder is a discrete linear time-invariant system. Every output of an encoder can be described by its own transfer function, which is closely related to a generator polynomial. An impulse response is connected with a transfer function through Z-transform.

Transfer functions for the first (non-recursive) encoder are:

- \( H_1(z) = 1 + z^{-1} + z^{-2}, \)
- \( H_2(z) = z^{-1} + z^{-2}, \)
- \( H_3(z) = 1 + z^{-2}. \)

Transfer functions for the second (recursive) encoder are:

- \( H_1(z) = \frac{1 + z^{-1} + z^{-3}}{1 - z^{-2} - z^{-3}}, \)
- \( H_2(z) = 1. \)

Define \( m \) by

\[
m = \max_i \text{polydeg}(H_i(1/z))
\]

where, for any rational function \( f(z) = P(z)/Q(z), \)

\( \text{polydeg}(f) = \max(\text{deg}(P), \text{deg}(Q)). \)

Then \( m \) is the maximum of the polynomial degrees of the \( H_i(1/z) \), and the constraint length is defined as \( K = m + 1 \). For instance, in the first example the constraint length is 3, and in the second the constraint length is 4.
4.6 TWO BASIC CONVOLUTIONAL ENCODER

- Convolutional code (2,1,2)

  n=2: 2 modulo-2 adders or 2 outputs

  k=1: 1 input

  M=2: 2 stages of shift register (K=M+1=2+1=3)

FIG. 4.6.1
Impulse Response for the Binary (2, 1, 2) Convolutional Code

\[ g_1 = (1, 1, 1, 0, \ldots) = (1, 1, 1), \quad g_2 = (1, 0, 1, 0, \ldots) = (1, 0, 1) \]
\[ v_1 = (1, 1, 1, 0, 1) \star (1, 1, 1) = (1, 0, 1, 0, 0, 1, 1) \]
\[ v_2 = (1, 1, 1, 0, 1) \star (1, 0, 1) = (1, 1, 0, 1, 0, 0, 1) \]

\( g \) stands for generator matrix, \( v \) is the output & \( u \) is input.

Convolutional code (3, 2, 1)

n=3: 3 modulo-2 adders or 3 outputs

k=2: 2 input

M=1: 1 stages of each shift register (K=2 each)
4.7 DECODING PROCEDURE

The most commonly used decoding algorithm for convolutional codes is the Viterbi Algorithm, which is a maximum likelihood sequence estimator (MLSE). The Viterbi algorithm is essentially a shortest path algorithm. The ML algorithm is too complex to search all available paths. End to end calculation is not feasible with the Viterbi algorithm. The Viterbi algorithm performs ML decoding by reducing its complexity. Eliminate the least likely trellis path at each transmission stage. Reduce decoding complexity with early rejection of unlike paths. Viterbi algorithm gets its efficiency via concentrating on survival paths of the trellis.
Fig 4.7.2 BER performance of soft and hard decoding
[5] ENCODER DESIGN PROCESS

5.1 HOW TO SIMULATE THE ENCODER

We can easily design the encoder model by the help of MATLAB software & also we are able to run the structure. Now we have to try to decrease the bit error rate by varying the internal values. The total procedure are given as follows.

The following model simulates this encoder.

To open the completed model, enter doc_convcoding at the MATLAB command line. To build the model, gather and configure these blocks:

- **Bernoulli Binary Generator**, in the Comm Sources library
  - Set Probability of a zero to .5.
  - Set Initial seed to any positive integer scalar, preferably the output of the randseed function.
  - Set Sample time to .5.
  - Check the Frame-based outputs check box.
  - Set Samples per frame to 2.

- **Convolutional Encoder**
  - Set Trellis structure to poly2trellis([5 4],[23 35 0; 0 5 13]).
• **Binary Symmetric Channel**, in the Channels library
  - Set Error probability to 0.02.
  - Set Initial seed to any positive integer scalar, preferably the output of the `randseed` function.
  - Clear the Output error vector check box.

• **Viterbi Decoder**
  - Set Trellis structure to `poly2trellis([5 4],[23 35 0; 0 5 13])`.
  - Set Decision type to Hard decision.

• **Error Rate Calculation**, in the Comm Sinks library
  - Set Receive delay to 68.
  - Set Output data to Port.
  - Check the Stop simulation check box.
  - Set Target number of errors to 100.

• **Display**, in the Simulink Sinks library
  - Drag the bottom edge of the icon to make the display big enough for three entries.

Connect the blocks as in the figure. From the model window's Simulation menu, select Configuration parameters. In the Configuration Parameters dialog box, set Stop time to inf.
5.2 OBSERVATION

Running the program we can observe that the output bit error rate is near about 2.5. By varying the puncturing value the output bit error rate can be minimized to 0.98. Finally we are successful to design errorless convolutional encoder.
5.3 A PERFECT CONVOLUTIONAL ARCHITECTURE

Fig-5.3.1
5.4 PUNCTURED COVOLUTIONAL CODES

Puncturing is a technique used to make a \( m/n \) rate code from a "basic" rate 1/2 code. It is reached by deletion of some bits in the encoder output. Bits are deleted according to \textit{puncturing matrix}. The following puncturing matrices are the most frequently used:

<table>
<thead>
<tr>
<th>Code rate</th>
<th>Puncturing matrix</th>
<th>Free distance (for NASA standard K=7 convolutional code)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2 (No perf.)</td>
<td>( \begin{bmatrix} 1 \ 1 \end{bmatrix} )</td>
<td>10</td>
</tr>
<tr>
<td>2/3</td>
<td>( \begin{bmatrix} 1 &amp; 0 \ 1 &amp; 1 \end{bmatrix} )</td>
<td>6</td>
</tr>
<tr>
<td>3/4</td>
<td>( \begin{bmatrix} 1 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 0 \end{bmatrix} )</td>
<td>5</td>
</tr>
<tr>
<td>5/6</td>
<td>( \begin{bmatrix} 1 &amp; 0 &amp; 1 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 0 &amp; 1 &amp; 0 \end{bmatrix} )</td>
<td>4</td>
</tr>
<tr>
<td>7/8</td>
<td>( \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 &amp; 1 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 1 &amp; 0 &amp; 1 &amp; 0 \end{bmatrix} )</td>
<td>3</td>
</tr>
</tbody>
</table>

For example, if we want to make a code with rate 2/3 using the appropriate matrix from the above table, we should take a basic encoder output and transmit every second bit from the first branch and every bit from the second one. The specific order of transmission is defined by the respective communication standard.

Punctured convolutional codes are widely used in the satellite communications, for example, in INTELSAT systems and Digital Video Broadcasting.

Punctured convolutional codes are also called "perforated".
5.5 DECODING ARCHITECTURE WITH PUNCTURING

**FIG 5.5.1**

**FIG 5.5.2**
[6] INDUSTRY APPLICATIONS

It is used widely in the field of communications system like WDMA, CDMA, 3G TECHNOLOGY, SATELLITE COMMUNICATION.
7.1 CONCLUSION

Through this project it has been notified that how to encode data using convolutional encoding method and how to minimize the error using convolution coding and also simulating and designing convolutional encoder using MATLAB for observing and minimizing bit error rate. Convolutinal codes is the most efficient & most realiable digital transmission process comparable to others.

7.2 FUTURE WORKS

Convolutional codes are used extensively in numerous applications in order to achieve reliable data transfer, including digital video, radio, mobile communication, and satellite communication. These codes are often implemented in concatenation with a hard-decision code, particularly Reed Solomon. Prior to turbo codes, such constructions were the most efficient, coming closest to the Shannon limit. Future work will include creating a friendly graphical user interface (GUI) for simulating the processes of convolutional coding and hard-decision / soft-decision Viterbi decoding as the results will be compared to the theoretical ones.
7.3 REFERENCES

This project has been brought into existence with the help of following links-
