ANALYZING THE DECODING PERFORMANCE OF RATE \( \frac{1}{2} \) CONVOLUTIONAL CODE

CH. RAVI KUMAR\(^1\) & K. PADMARAJU\(^2\)

\(^1\)Assistant Professor, Department of E. C. E, Sir C. R. Reddy College of Engineering, Eluru, Andhra Pradesh, India
\(^2\)Principal, JNTU Kakinada, Kakinada, Andhra Pradesh, India

ABSTRACT

The idea behind a convolutional code is to make every codeword symbol be the weighted sum of the various input message symbols. In telecommunication, a convolutional code is a type of error-correcting code in which. In this paper we are analyze decoding performance, for different ‘K’ values and discuss the performance of hard decision decoding and soft decision decoding. We use optimization algorithm to reduce the decoding delay in viterbi decoding approach. So a method is proposed for fast decoding of convolutional codes which reduces decoding delay using particle swarm optimization. In soft decision decoding the performance is improvement of approximately 2db in required signal-to-noise ratio compared to the hard decision decoding.

KEYWORDS: Convolutional Code, Constraint Length, Particle Swarm Optimization, Shannon’s Limit, Viterbi Decoding

INTRODUCTION

In coding techniques i.e block codes and convolutional codes Shannon’s channel capacity theorem plays an important role that related channel capacity vs signal band with. capacity. Coding techniques are used in reliable communication, by obtaining noise, fading, and interference etc [1]. in terms of bandwidth/power efficiency, its codeword or constraint length should be increased to such an intolerable degree that the maximum likelihood decoding can become unrealizable[2]. The capacity of the channel represented in bits per second and the signal band with in Hz and errors are caused by random noise [3] can be given the equations (1) and (2) below.

\[
c = \log\left(1 + \frac{E_b}{N_0}\right) \text{ bits/sec}
\]  

(1)

where \( S \) is signal power (W), \( N \) is noise power (W), and \( N_0 \) is noise power spectral density (W/Hz). Here, if band with of the signal is infinity, the channel capacity would be approach the constant value given below.

\[
\frac{E_b}{N_0} = \frac{1}{\log_2 e} = 0.693
\]

(2)

In order to achieve this bound, i.e. \( Eb/N0 \approx -1.59 \) dB value, it would be necessary to use a code with a such long length that encoding and decoding would be practically impossible. However, the most significant step in obtaining this target, was by Forney, who found that a long code length could be achieved by concatenation of two simple component codes with short lengths linked by an interleaver. Channel coding refers to the class of signal transformations designed to improve communication performance by enabling the transmitted signals to better withstand the effects of various channel impairments, such as noise, interference, and fading, for error free
transmission or reliability in transmission bring the transmission rate close to Shannon’s channel capacity theorem [4]. We use the model for a binary input AWGN channel, given by

\[ r_l = a_l + n_l \]  \hspace{1cm} (3)

Where \( r_l \) is the l-th received symbol and \( a_l \) is the l-th symbol to be transmitted, and the received symbol corrupted by noise with zero mean and variance, the noise signal have noise power spectral density i.e \( \text{N}_0/2 \). The additive white Gaussian noise (AWGN) channel represented in equation (3) has been a powerful relation among received signal and transmitted signal. In general communication concepts as satellite, line-of-site terrestrial communication use AWGN channels. In this present work we use 1/3 rate convolutional encoder and decoder to mitigate the results such as Shannon’s channel capacity theorem. In figure 1 shows the relation between uncoded and coded performance with related Shannon’s channel capacity theorem and there is coding gain.

![Figure 1: A Simulated BER (In Log Scale) Versus Eb/N0 (In Db) Curve](image)

To analyze this concept, in this paper we consider convolutional code of rate \( \frac{1}{3} \) for a binary input, and the channel is additive white Gaussian noise channel having zero mean and variance. In convolution code the constraint length influence the coding gain, in specific rate \( 1/3 \) the code word length of 3 binary bits for each single in put binary data. According to this definition, The constraint length equals nk, where n is number of modulo-2 adders.[5].

Convolutional coding is done by combining fixed number of input bits stored in fixed length shift registers and are combined with the help of modulo-2 adders. This operation is equivalent to convolution so it is called convolutional coding [6]. There are three decoding algorithms namely viterbi, sequential and maximum a posterior (MAP). in this paper use viterbi algorithm that is closes to received sequence [7].

After convolutional encoding and decoding, we analyze the performance and seek to optimize using optimization method i.e particle swam optimization is used in this paper the principle used in optimization method are velocity of swam that travel and hunting for the food [8].

**CONVOLUTIONAL CODING**

Convolutional code is a type of error-correcting code in which each \( m \)-bit information symbol (each \( m \)-bit string) to be encoded is transformed into an \( n \)-bit symbol, where \( m/n \) is the code rate \( (n \geq m) \) and the transformation is a function of the last \( k \) information symbols, where \( k \) is the constraint length of the code[9]. 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.

In this work, we propose to use a 1/n, more specifically, a 1/3 convolution encoder/decoder, to mitigate the disturbance resulting from such a channel. However, before doing so, let us revisit some of classical coding techniques presented in this class, and motivate our reasoning for choosing a convolution code. An encoder can be linear time invariant filter with filter banks $g_i$, where $i$ is integer value $1$ to $n$ and the message bits given to the convolutional encoder given in figure 2 where the output code word $c_i=m(D) g_i(D)$. Hence encoding schemes are multiple we can design an multiple encoder to achieve rate 1/3 with the memory element related to constraint length $K$.

![Figure 2: The General Form of 1/N Convolution Encoder](image)

**ENCODING SCHEMES**

The 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:

- $n_1 = m_1 + m_0 + m_{-1}$
- $n_2 = m_0 + m_3$
- $n_3 = m_1 + m_3$.

A convolutional encoder with the rate $R = m/n$ is constructed on a basis of $m$ input bits, $n$ output bits and $k-1$ memory units, where $k$ is the constraint length. The memory outputs and input data are joined each other in the required combination by an Exclusive OR (XOR) operator which generates the output bits.

In the convolutional encoder, one bit entering the encoder will affect to the code performance for $m+1$ time slots, which represents the constraint length value of the code. Since an XOR is a linear operation, the convolutional encoder is a linear feed forward circuit. Based on this property, the encoder outputs can be obtained by convolution of input bits with $n$ impulse responses. The impulse responses are obtained by considering input bit stream $(100...0)$ and observing the output sequences. Generally, these impulse responses are called generator sequences having lengths equal to the constraint length of the code. The generator sequences determine the existence connection between the encoder memories, its input and output.

Different encoding schemes are implemented depends on the constraint length $K$ for a 1/3 rate convolutional system. Consider with $K=3$, 4, 5, 6. Hence the code word is generated i.e $C$ for the respective message $m$ then the code
word is transmitted through a AWGN channel model given by equation 3 then we need to recover by decoding algorithm.

Table 1: Rate 2/3 Convolution Code with Minimum Distance [10].

<table>
<thead>
<tr>
<th>K</th>
<th>G0</th>
<th>G1</th>
<th>G2</th>
<th>D Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0110</td>
<td>0000</td>
<td>1000</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>100100</td>
<td>101011</td>
<td>111101</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>1001</td>
<td>1110</td>
<td>1110</td>
<td>16</td>
</tr>
</tbody>
</table>

Figure 3: Optimal Rate 1/3 Convolutional Encoder for K = 4

DECODING SCHEMES

The decoding algorithm can be designed by two methods one is hard decision decoding and another is soft decision decoding, and we analyze the performance of both hard decision decoding and soft decision decoding. In the hard decision decoding the received information from the noisy channel is quantized to 2 levels producing the binary data for the decoder input. In this approach, finding the most likelihood path is conducted based on selection of paths with the minimum Hamming distance. Hamming distance is defined as the number of different bits between the decoded data and the input data of the encoder. For example, due to the difference in the third and forth bits of data (101001) and (100101), Hamming distance is equal to 2. In the Viterbi algorithm this distance is introduced as branch metric, which for simplicity is called just metric. At each time unit of the trellis diagram, branch metrics are computed based on how many bits are different between the received information and the transmitted information. Then, the path with the minimum branch metric is selected as the survivor path. The obtained branch metric at each time unit is added to the previously accumulated branch metric forming the path metric at the relevant time unit. This procedure is continued to decode all the received information. Following, an example is presented to clarify this approach [12].

• HARD DECODING

The branch metric is the chosen parameter for decoding in trellis map, consider hamming distance among received code words. The demodulator output can be configured in a variety of ways. It can be implemented to make a firm or hard decision as to whether received represents zero and one. In this case the output of the demodulator is quantized to two levels, zero and one, and fed in to the decoder. Since the decoder operates the hard decisions made dy the demodulator,
the decoding is called hard decision decoding.

- **SOFT DECODING**

  The demodulator can also be configured to feed the decoder with a quantized value of greater than two levels, such as implementation furnishes the decoder with more information than is provided in hard decision case. When the quantized value of greater than two, the decoder is called soft decision decoder. For a Gaussian channel, eight level quantization results in performance improvement of approximately 2 dB in required signal-to-noise ratio compared to two level quantization. This means that eight level soft decision decoding can provide the same probability of bit error as that of hard decision decoding, but requires 2 dB less $E_b/N_0$ for the same performance.

**PARTICLE SWARM OPTIMIZATION**

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart in 1995. PSO is similar to Genetic Algorithm (GA) in that the system is initialized with a population of random solutions. It is unlike a GA, however, in that each potential solution is also assigned a randomized velocity, and the potential solutions, called *particles*, are then “flown” through the problem space. Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (This fitness value is also stored) This value is called *pbest*. Another “best” value that is tracked by the *global* version of the particle swarm optimizer the overall best value, and its location, obtained so far by any particle in the population. This location is called *gbest* [14]. The particle swarm optimization concept consists of, at each time step, changing the velocity (accelerating) each particle toward its pbest and gbest locations (global version of PSO). Acceleration is weighted by a random numbers being generated for acceleration toward pbest and gbest locations. This is also a *local* version of PSO in which, in addition to pbest, each particle keeps track of the best solution, called lbest, attained within a local topological neighborhood of particles.

The (original) process for implementing the global version of PSO is as follows:

- Initialize a population (array) of particles with random positions and velocities on $d$ dimensions in the problem space.
- For each particle, evaluate the desired optimization fitness function in $d$ variables.
- Compare particle’s fitness evaluation with particle’s *pbest*. If current value is better than pbest, then set pbest value equal to the current value and the pbest location equal to the current location in d-dimensional space.
- Compare fitness evaluation with the population’s overall previous best. If current value is better than gbest, then reset gbest to the current particle’s array index and value.
- Change the velocity and position of the particle according to equations (1) and (2), respectively:
- Loop to step 2) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations (generations).

**PSO ALGORITHM**

www.tjprc.org  
editor@tjprc.org
The number of particles typical range is 20-50. 10 particles are enough for good results. For some difficult or special problems, 100-200 particles can be used. Range of particles is determined by the problem to be optimized. \( V_{\text{max}} \) determines the maximum change one particle can take during one iteration. Stop condition is the maximum number of iterations the PSO execute and the minimum error requirement. Fitness function is supplying the candidate solution to the objective function. Update the individual and global best and fatnesses by comparing the newly functions. The PSO algorithm given in figure 4 the velocity of each particle is updated using the equation below.

\[
V_i(t) = w \cdot V_i(t) + c_1 \cdot r_1 \cdot [p_i(t) - X_i(t)] + c_2 \cdot r_2 \cdot [g(t) - X_i(t)] \\
X_i(t+1) = X_i(t) + V_i(t+1)
\]

where \( V_i(t) \) and \( X_i(t) \) are velocity and position of the particle at time \( t \) and \( p_i(t), g(t) \) are the particle best position, Pbest and global best position, Gbest respectively. The parameters \( w \in [0, 1.2], c_1 \) and \( c_2 \in [0, 2] \) are user supplied coefficients and \( r_1, r_2 \in [0, 1] \) are random value regenerated for each velocity update.

Based, among other things, on findings from social simulations, it was decided to design a “local” version of the particle swarm. In this version, particles have information only of their own and their neighbour’s bests, rather than that of the entire group. Instead of moving toward a kind of stochastic average of \( p_{\text{best}} \) and \( g_{\text{best}} \) (The best location of the entire group), particles move toward points defined by the \( p_{\text{best}} \) and “lbest,” which is the index of the particle with the best evaluation in the particle’s neighbourhood [15].

**PSO FOR CONVOLUTIONAL CODE OPTIMIZATION**

As complexity of Viterbi decoding algorithm increases exponentially with constraint length, it is adopted to decoding of shorter convolutional codes. Fast decoding of convolutional codes is possible using particle swarm optimization algorithm. This algorithm reduces the searching area in the trellis decoding and shortens decoding delay and this algorithm also reduces bit error rate [17]. Convolutional code optimization using particle swarm optimization is done using following steps.

Figure 4: PSO Algorithm
• **Generate Polynomial:** The polynomial description of convolutional encoder describes the connection among shift registers and modulo-2 adders. Form a binary representation by placing a 1 in each connection line from shift feed into the adder and 0 elsewhere. Convert this binary representation into octal representation.

• **Draw the Trellis:** A trellis description of a convolutional encoder shows how each possible input of encoder influences both the output and state transition of encoder. Start with a polynomial description of the encoder and use `poly2trellis` function to convert it to valid structure.

• **Calculate Bit Error Rate:** Calculate bit error rate using octal code and trellis structure. To decode convolutional code, use the `vitdec` function with the flag `hard` and with binary input data. Because the output of `convenc` is binary, hard decision decoding can use the output of `convenc` directly. After white Gaussian noise (AWGN) is added to the code.

• **Update Particle’s Position and Velocity:** At each time, all particles have an update. At iteration $t$, the $t_{th}$ element in the vector is updated. Particle’s position is decided by velocity as in equation (4). At the decoding process, the update of velocity and location must act up to transfer rule of encoder state. Select lowest value of bit error rate as fitness function.

• **Update Personal Best Position and Global Best Position:** Update personal best position and global best position after all particles position have been updated.

• **Ending Condition:** When iteration $t=\text{L}$, all particle’s position have been updated for $\text{L}$ times and reaches the end.

These steps are clearly illustrated in the following flowchart shown in figure 5.

![Flowchart of Convolutional Encoder Using PSO](image)

**Figure 5: Convolutional Encoder Using PSO**

**RESULTS**

The performance of particle swarm optimization with convolutional code is verified using MATLAB software. The results show that using particle swarm optimization for convolutional code reduces bit error rate and decoding delay.
We perform experiments for different constraint lengths of rate ⅓ convolutional code for hard and soft Viterbi decoding and optimize using particle swarm optimization. Experiments are done with 100 trials and the results are briefly discussed in the following tables table II and table III and are shown in figures from figure 6 to figure 13. They are obtained with less decoding delay and also good bit error rates are achieved within less number of trials. Without optimization, to obtain such results 1000 trials or more are required which consumes large time. The time taken to obtain the simulation result of each scheme is shown below in table II and table III along with the BER values corresponding to \( E_b/N_0 \) (dB) values.

### Table 2: Decoding Performance Using PSO for ⅓ Rate Convolutional Hard and Soft Decoding

<table>
<thead>
<tr>
<th>K</th>
<th>Convolutional Code, Hard Decision Decoding</th>
<th>Convolutional Code, Soft Decision Decoding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( E_b/N_0 ) (db)</td>
<td>BER</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>( 10^{-1} )</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>( 10^{-4} )</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>( 10^{-5} )</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>( 10^{-6} )</td>
</tr>
</tbody>
</table>

Figure 6: Simulation Result of K=3 Hard Decoding

Figure 7: Simulation Result of K=3 Soft Decoding
Analyzing the Decoding Performance of Rate ½ Convolutional Code

Figure 8: Simulation Result of K=4 Hard Decoding

Figure 9: Simulation Result of K=4 Soft Decoding

Figure 10: Simulation Result of K=5 Hard Decoding

Figure 11: Simulation Result of K=5 Soft Decoding
CONCLUSIONS

The Viterbi decoding algorithm used for decoding of convolutional codes has decoding delay. In this paper use particle swarm optimization algorithm to reduce decoding delay. Convolutional code of rate $\frac{1}{3}$ for different constraint lengths, for hard and soft Viterbi decoding is optimized using PSO. It is observed from the results that the algorithm gives better bit error rates and in soft decision decoding the performance is improvement of approximately 2db in required signal-to-noise ratio compared to the hard decision decoding.

REFERENCES


Analyzing the Decoding Performance of Rate ⅓ Convolutional Code


AUTHOR’S DETAIL

Ch. Ravikumar received B. E degree in Electronics and communication Engineering from Andhra University, M. Tech degree in communications from JNTUK, Kakinada, pursuing Ph. D from JNTUK, Kakinada. He has worked as 1 year Apprentice trainee engineer in BDL, signal condition Systems, Hyderabad. He has 9 years of teaching experience and is Assistant Professor of ECE, Sir C R Reddy engineering college Eluru, India. He has published 15 research papers in National/ International Journals/ Conferences and guiding 10 M-Tech students in the areas of Communications, Antennas.
K. Padma Raju received B. Tech from Nagarjuna University, M. Tech from NITW, Ph. D from Andhra University, India in 1989, 1992 & 2005 respectively and Post Doctoral Fellowship at Hoseo University, South Korea in 2007. He has worked as Digital Signal Processing engineer in Signion Systems, Hyderabad, India before joining Jawaharlal Nehru Technological University Kakinada, India. He has 20 years of teaching experience and is Professor of ECE, JNTUK, India. Currently he is working as Principal, College of engineering, JNTUK. Prior to this, he has worked as IIIP&T Director for JNTUK. He has also worked as a Professor at Hoseo University, South Korea during 2006-2007. He has published 40 research papers in National/ International Journals/ Conferences and guiding 10 research students in the areas of Antennas, Smart grids, and Communications. He is Fellow of IETE & IE and Life Member of ISTE & SEMCE.