Error performance analysis of forward error correction using convolutional encoding in the presence of (1/f) noise

Yasin Yousif Al-Aboosi, Ammar Ali Sahrab, Amal Ibrahim Nasser, Hussein A. Abdulnabi
Department of Electrical Engineering, College of Engineering, Al-Mustansiryah University, Baghdad, Iraq

Article Info

Article history:

Received Dec 15, 2022
Revised Jan 20, 2023
Accepted Feb 14, 2023

Keywords:

Bit error rate
Colored noise
Convolution code
Forward error correction
Viterbi decoder

ABSTRACT

Any communication scheme's principal goal is providing error-free data transmission. By increasing the rate at which data could be transmitted through a channel and maintaining a given error rate, this coding is advantageous. The message bits to be transmitted will gradually receive more bits thanks to the convolution (channel) encoder. At the receiver end of the channel, a Viterbi decoder is utilized in order to extract original message sequence from the received data. Widely utilized error correction approaches in communication systems for the enhancement of bit error rate (BER) performance are Viterbi decoding and convolutional encoding. The Viterbi decoder and convolution encoder rate for constraints with lengths of 2 and 6 and bit rates of 1⁄2 and 1⁄3 are shown in this study in the presence of (1/f) noise. The performance regarding the convolutional encoding/hard decision Viterbi decoding forward error correction (FEC) method affects the simulation outcomes. The findings demonstrate that the BER as function of signal to noise ratio (SNR) acquired for uncoded binary phase shift keying (BPSK) with the existence of additive white Gaussian noise (AWGN) is inferior to that acquired with the use of a hard decision Viterbi decoder.

Corresponding Author:

Yasin Yousif Al-Aboosi
Department of Electrical Engineering, College of Engineering, Al-Mustansiryah University
Baghdad, Iraq
Email: alaboosiyasin@gmail.com

1. INTRODUCTION

Data transmission through noisy channels is addressed by coding theory, which employs a variety of source and channel coding/decoding algorithms. Forward error correction (FEC) schemes frequently employ convolutional coding. The widely utilized Viterbi algorithm serves as the foundation for the primary decoding method for convolution codes. Convolutional encoding with the Viterbi decoding represents potent FEC approach which is especially well-suited to a channel where transmitted signal is mostly contaminated by additive white Gaussian noise (AWGN) in conventional systems [1]. However, in fact, colored noise is present [2]. Through carefully designing redundant information to be added to data that is being transmitted via the channel, FEC aims to increase the channel's capacity [3]. For the most part, Viterbi algorithm conducts maximal likelihood decoding to fix errors in received data that is brought on by channel noise. Therefore, reduce bit error rate (BER) to enhance performance. Because Viterbi decoding has set decoding time, it’s advantageous for the implementation of hardware decoder. The specifications for Viterbi decoder, a processor that executes the Viterbi algorithm, vary depending on the application. Viterbi algorithm uses the greatest resources and is the most effective [4]–[6]. In Figure 1, a general block architecture regarding a digital communication system with the use of convolution encoder and Viterbi decoder has been shown in the presence of colored noise.

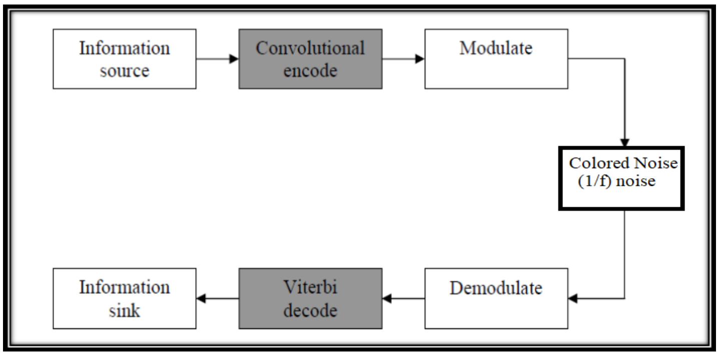

Figure 1. Diagram of the digital communication system utilizing convolution encoder and Viterbi decoder in the presence of the colored noise

2. METHOD

Shift registers are used in fixed numbers to create convolutional encoders. The encoder output is created through adding bits in shift register, which each input bit enters. The quantity of modulo 2-adders that are utilized in combination with shift registers specifies the number of the output bits [1], [4], [7].

2.1. Encoder parameters

Three parameters are frequently used to describe convolutional codes: n, k, and m, in which n represent number of the output bits, m represent number of the memory registers and k represent number of the input bits. The code rate, which has been denoted by k/n, is a measurement of a code's bandwidth effectiveness. Except for the applications of deep space, when code rate values as low as 1/100 or longer could be used, k and n parameters typically vary between 1 and 8, code rate between 1/8 and 7/8, and m between 2 and 10. The parameters (n, k, and L) could also be used to specify convolutional codes. L is referred to as constraint length of code and represents the amount of bits in an encoder memory which have an impact on how the n output bits are generated. Instead of (n, k, and m) codes, convolution codes that have been presented here will be denoted as (n, k, and L) codes [6], [7].

2.2. Structure of the encoder

The connections of the encoder have been described by term generator polynomial (g). The generator polynomial for that output bit refers to decision of which bits (in memory registers) should be added (with the use of modulo-q adders) for producing the output bits. For any m order code, there are a variety of polynomials to choose from. In most cases, using computer simulations, good polynomials are discovered by trial and error. The next presumptions were established for better understanding a convolutional encoder's working and the FEC method [8], [9]: i) A (2, 1, 3) convolutional encoder has been utilized, ii) a three-bit sequence of input has been utilized by bits [1 0 1], and iii) 2 generator polynomials have been utilized, which were specified by bits [1 0 1] and [1 1 1].

Constructing convolutional encoder is simple. Initially, m boxes are drawn in order to represent m memory registers. Next, the n output bits by n modulo-2 adders are drawn. Lastly, we use the bits defining generator polynomials for connecting the memory registers to adders. A (2, 1, 3) convolutional encoder has been depicted in Figure 2. The 2 generator polynomials that have been designated by bits [1 0 1] and [1 1 1] will be utilized for encoding the three-bit input sequence [1 0 1] using this encoder. The output bits 1 and 2 are represented by v1 and v2, respectively, whereas u1 represents the input bit. The memory registers' initial states, initially set to the value of 0, are represented by u-1 and u-0.

2.3. Convolutional representation

The convolutional encoder's logical operation was shown in the preceding section. A look-up table, also known as table of state transitions, could be used by the convolutional encoder to do the encoding. There are four items in the state transition table: i) input bit, ii) encoder state that is 1 of 4 potential states (00 01 10 11) for (2, 1, 3) convolutional encoder, iii) output bits that for (2, 1, 3) convolution encoder are: 00 01 10 11, due to the fact that only 2 bits would be output, and iv) output state which represents the following bit’s input state. The output state which is state transition table for (2, 1, 3) convolutional encoder is as it has been clarified in Table 1. There are graphically 3 ways for the representation of encoder for the purpose of gaining a better understanding concerning its operation [10], [11]. Those include: i) tree diagram representation and ii) state diagram representation.

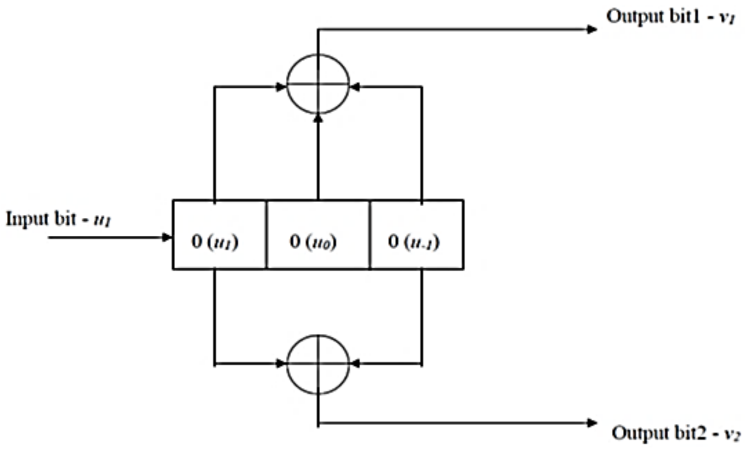

Figure 2. A (2, 1, 3) convolutional encoder

Table 1. State transitions for (2, 1, 3) convolutional encoder

Input bit	Input state	Output bits	Output state
0	00	00	00
1	00	11	10
0	01	11	00
1	01	00	10
0	10	10	01
1	10	01	11
0	11	01	01
1	11	10	11

2.3.1. Representation of the state diagram

The state diagram for (2, 1, 3) convolution encoder is as has been depicted by Figure 3. Contents of two shift register cells have been referred to as a state. A state is represented by each circle. The encoder can be found in one of such states at any one time. As bits arrive, lines from and to it display possible state transitions. The arrival of bit 1 or the arrival of bit 0 are the only 2 cases that can occur simultaneously. The encoder is able to "jump" into a different state as a result of each of these two events. As can be seen, the state diagram contains the same data as the state transition table but does it graphically. The dotted lines exhibit arrival of bit 1, while solid ones show arrival of bit 0. Each case's output bits are displayed on a line, and an arrow points to the state transition [12], [13].

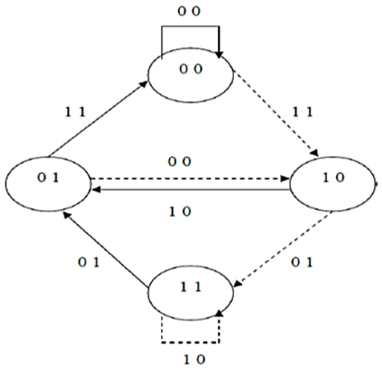

Figure 3. Diagram for (2, 1, 3) convolutional encoder

2.3.2. Tree diagram representation

Figure 4 displays a tree diagram for the convolutional encoder (2, 1, 3). State diagram entirely describes the encoder, however it's difficult to utilize it to monitor encoder transitions over time. Due to the fact that the diagram does not depict time history, this is obvious. The tree diagram gives the state diagram a time dimension by depicting time passage as we delve farther to the 3 branches. Which makes it superior method for describing convolutional codes compared to the state diagram. With the tree diagram, branches of a tree are travelled based upon arrival of bit 1 or bit 0, rather than "jumping" from one state to another. In the encoder's input, if bit 0 is received, we move up the tree, and if bit 1, we move down. The output state is represented by the bits in parenthesis after the first two bits, which stand for the output bits [14], [15].

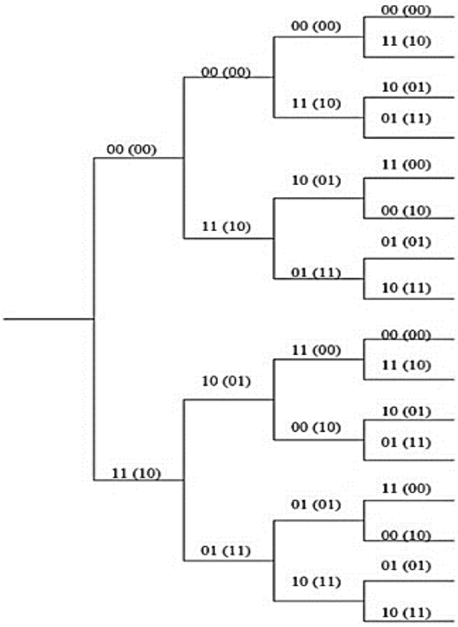

Figure 4. Tree diagram for (2, 1, 3) convolutional encoder

3. VITERBI DECODING

The process of the recovery of encoded input data stream at receiver after it was transmitted over a channel has been referred to as the channel decoding. Sequential decoding and maximum likelihood decoding, which is usually referred to as the Viterbi decoding, represent the 2 main channel decoding types for convolutional codes. Maximum likelihood decoding, which is often known as Viterbi decoding, has been created by Viterbi, a founding member of Qualcomm corporation. Error bounds for the convolutional codes and asymptotically optimal decoding algorithm is his seminal study on the method. Viterbi decoding is more suitable for hardware implementation because, unlike sequential decoding, decoding time is fixed rather than variable. Here, we methodically eliminate alternatives at each trellis level [16], [17]. The following criteria were utilized in order to narrow the options:

- Errors happen in an infrequent manner. Error likelihood has been found small
- The likelihood of 2 consecutive errors is considerably smaller when compared to one error that is the errors are distributed in a random manner

The entire received sequence of some specific length is examined by Viterbi decoder, which calculates a metric for every one of the paths and bases its decisions on that metric. Up until 2 paths meet at one node, all paths are taken. One of the two paths is subsequently chosen depending on an earlier decision. The chosen paths are referred to as the survivors. There are a total of 2N potential received sequences for an N-bit sequence. Only 2kL of these are valid. Just one survivor path is selected depending on a decision when two paths converge on a single node. There are two ways to make this decision, leading to the 2 Viterbi decoding forms [18]: i) hard and ii) soft decision Viterbi decoding.

3.1. Trellis explanation

The hard decision Viterbi decoder's actual working is depicted in the following figure. Depending on the example taken, a trellis is drawn for each time tick of decoded three-bit input stream [1 0 1]. The hard decision Viterbi decoder assumes that corrupted data bit stream at input is [01 10 00 10 11]. The path that has the maximum metric has been sought out and a winner path is traced after five stages. The decoded sequence is represented in Figure 5 and follows the path made up of states 00, 10, 01, 10, 01, 00, which correspond to the bit values 10100. It can be seen the way that the hard decision Viterbi decoder generates decoded data bit stream from convolutional encoded input data bit stream transmitted through a coloured noisy channel from transmitter by applying maximum Hamming distances [19].

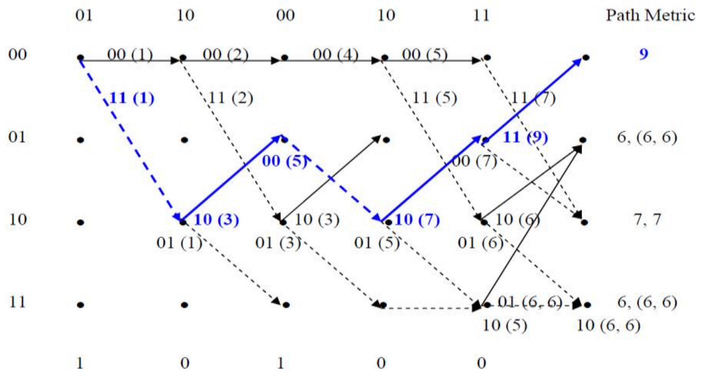

Figure 5. Decoded sequence 10100 for noisy encoded bit stream 01 10 00 10 11

4. COLORED NOISE GENERATION USING MATLAB

In various applications, white noise with a Gaussian distribution is assumed. The definitions of the autocorrelation and power spectrum are [12]:

$$r_{\epsilon\epsilon}[m] = E\{\epsilon[m]\epsilon[m+k]\} = \sigma_{\epsilon}^{2}\delta[m] \tag{1}$$

$$S_{\epsilon\epsilon}\left[e^{j2\pi f}\right] = F.T._{m\to f}\ \{r_{\epsilon\epsilon}[k]\} = \frac{N_0}{2}\frac{-f_s}{2} \le f \le \frac{f_s}{2} \tag{2}$$

The power spectrum density (PSD) of the white Gaussian noise represents constant value over full range of the frequencies, all of the frequency values range with no/2 magnitude. For some certain time instant, it had shaped the probability distribution function pdf $\rho_{\epsilon}(\epsilon)$ that has been given as (3) [20]:

$$\rho_{\epsilon}(\epsilon) = \frac{1}{\sigma_{\epsilon}\sqrt{2\pi}}e^{-\frac{(\epsilon-\mu_{\epsilon})^2}{2\sigma_{\epsilon}^2}} \tag{3}$$

in which the mean value is represented by $\mu\epsilon$ and the standard deviation by $\sigma\epsilon$. Since all of the samples are Gaussian and share same statistical characteristics, delta function on autocorrelation functions indicates that nearby samples are not dependent. The observed samples have been thought to be independently identically distributed (i.i.d.). Figures 6 and 7 display the AWGN's time representation, PSD, respectively [21]. Figure 8 shows the autocorrelation function for AWGN and its clear similar to delta function.

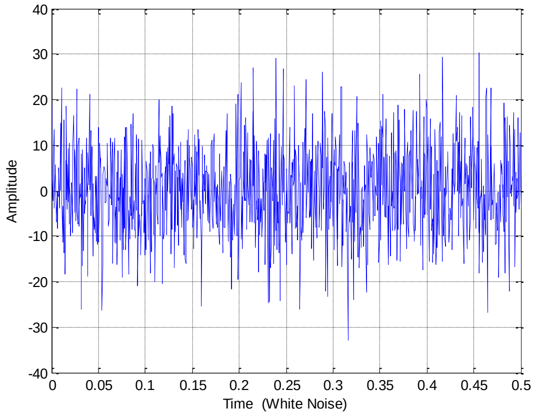

Figure 6. Time representation of the AWGN

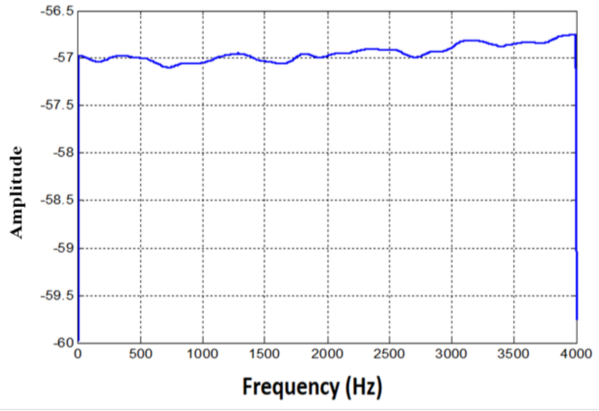

Figure 7. Power spectral density of the AWGN

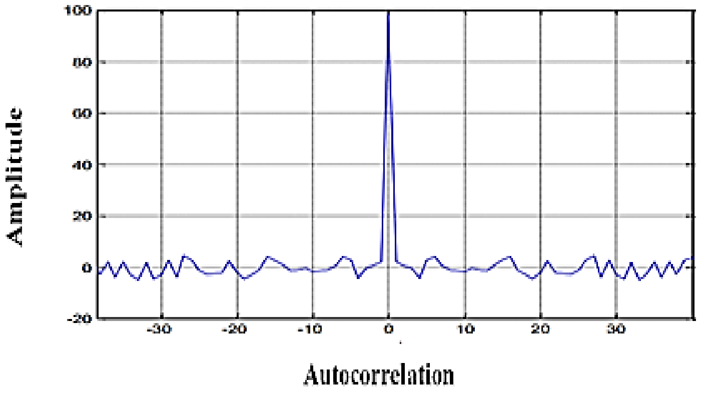

Figure 8. Autocorrelation function of the AWGN

According to Figure 9, white Gaussian noise is processed through a low-pass filter to produce color noise. It is coloured with 1st order low-pass Butterworth filter with cut-off frequency of 0.1 fs and added to signal of interest. This filter represents all-pole filter. The equation: gives its magnitude squared frequency response as (4) [22], [23]:

$$|H(w)|^2 = \frac{1}{1+({}^{w}/_{w_c})^{2N}} \tag{4}$$

in which w represent the digital frequency, w (c) represent the 3 dB cut-off digital frequency, N represent the filter order. Consequently, the frequency response of a first order Butterworth filter is proportional to (5):

$$|H(w)|^2\ \alpha \frac{1}{w^2} \tag{5}$$

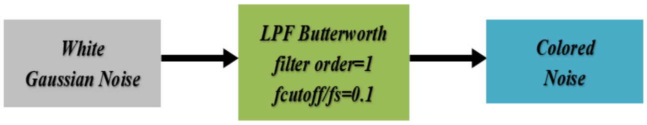

Figure 9. Generation of colored noise from white noise

Figure 10 shows the time representation of generated colored noise obviously the difference with the Figure 6. The PSD regarding the generated colored noise is shown in Figure 11; it is evident that the noise isn’t constant across the entire range of the frequencies and that it is proportional to 1/f noise. The autocorrelation function related to the colored noise is illustrated in Figure 12. The autocorrelation functions' sinc function indicates that nearby samples are dependant when all of the samples are Gaussian and have the same statistical characteristics [24], [25].

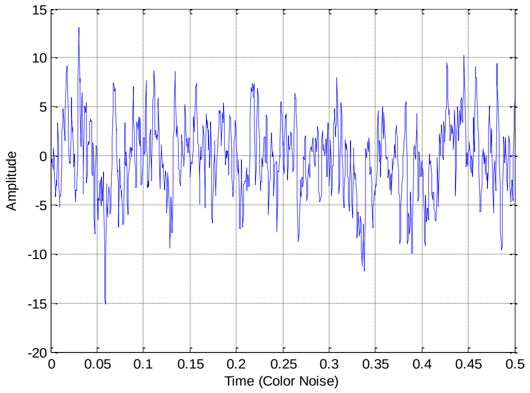

Figure 10. Time representation of the colored noise

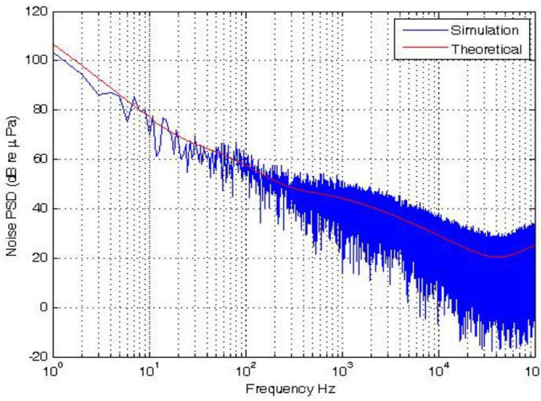

Figure 11. PSD of the colored noise

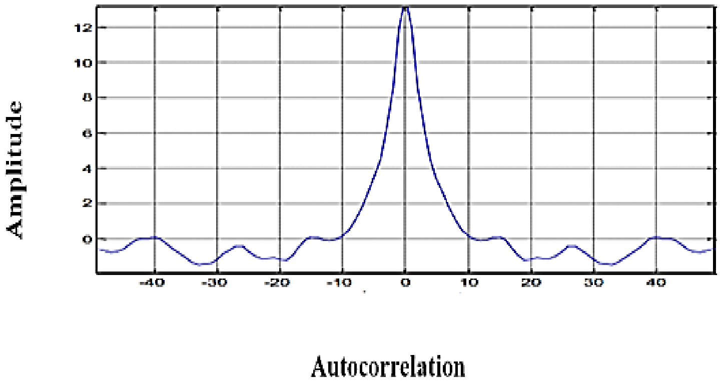

Figure 12. Autocorrelation function of the colored noise

5. RESULTS

The performance regarding the convolutional encoding/hard decision Viterbi decoding FEC method affects the simulation results. Those performance factors aid in the investigation of how the FEC method might be used in practical settings. The next performance factors are taken into account: i) encoder memory size and ii) signal to noise ratio (SNR). A hard decision Viterbi decoder with 1/2 code rate and memory m=3 is used to display the BER as an SNR function in Figure 13. SNR (Eb/No) in dB obtained with our MATLAB implementation utilizing the collection of common polynomials is shown in Table 2.

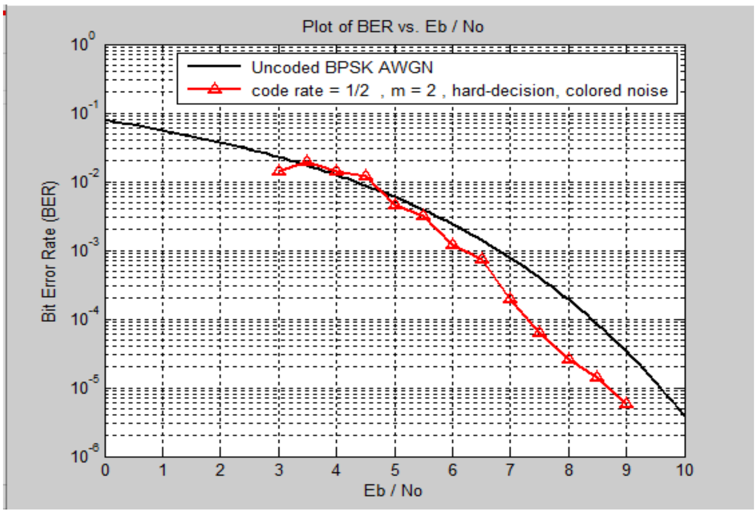

Figure 13. BER plot versus Eb/No for a rate 1/2, m=3 encoder, with the consideration of hard decision Viterbi decoder

Table 2. SNR (Eb/No) in dB for the rate 1/2, m=3 encoder

BER	Uncoded BPSK	Coded
10^{-2}	3.5	3.5
10^{-3}	7.1	6
10^{-4}	8.3	7.3

The performance of a hard decision Viterbi decoder with a 1/3 code rate and memory m=3 is depicted in Figure 14. The SNR (Eb/No) in dB obtained with the proposed MATLAB implementation utilizing common polynomials is shown in Table 3. The performance related to a hard decision Viterbi decoder with a 1/2 code rate and memory m=6 is depicted in Figure 15. SNR (Eb/No) in dB that had been obtained with our MATLAB implementation utilizing common polynomials is shown in Table 4.

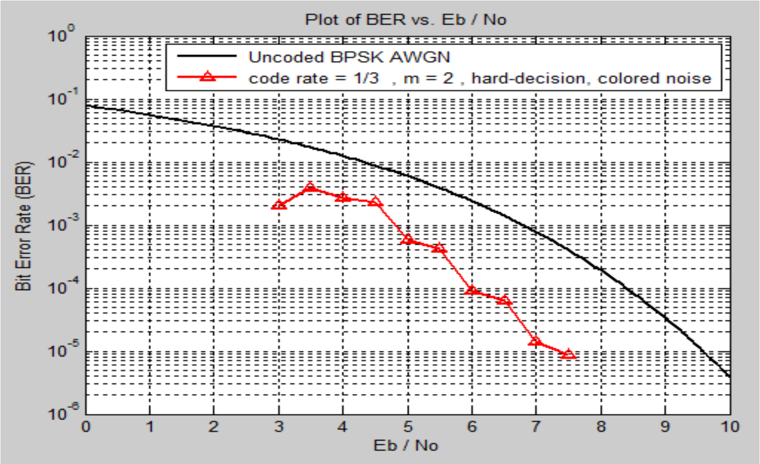

Figure 14. Plot of BER versus Eb/No for a rate 1/3, m=3 encoder, taking under consideration a hard decision Viterbi decoder

Table 3. SNR (Eb/No) in dB for rate of 1/3, m=3 encoder

BER	UnCoded BPSK	Coded
10^{-3}	7	4.8
10^{-4}	7.4	6
10^{-5}	9.2	6.8

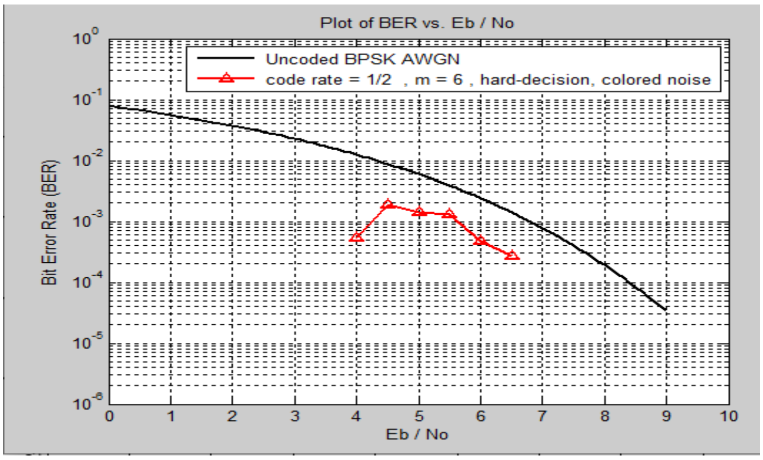

Figure 15. BER Plot vs. Eb/No for a rate 1/2, m=6 encoder, taking under consideration a hard decision Viterbi decoder

Table 4. SNR (Eb/No) in dB for 1/2 rate, m=6 encoder

BER	UnCoded BPSK	Coded
$10^{-}3$	7	4.2
10^{-4}	8.3	6.8

6. CONCLUSION

The design of a Viterbi decoder and a convolutional encoder, which could encode a digital data bit stream and produce a code word which could be transmitted and decoded at the destination. For each constraint length, rates of 1/2 and 1/3 were used when designing the encoder, which had constraints of lengths 2 and 6. The Viterbi decoder has been engineered for the determination of decoding path with least design of metrics to be sent to the port of the decoder output. Design of a Viterbi decoder and convolutional encoder with the use of MATLAB has proven successful, with results in terms of BER vs SNR in the presence of colored noise. Future work could apply Turbo codes and Reed-Solomon coding for the signal corrupted by colored noise using the FEC approach.

REFERENCES

[1] B. Sklar, *Digital communications: fundamentals and applications*. Los Angeles: Prentice Hall, 1988.

[2] A. A. Zali, A. Z. Sha'ameri, Y. Y. Mohammed, G. Hussin, and C. Y. Mei, "Enhancement of power measurement by decimation in 1/f noise," in *2015 IEEE Student Conference on Research and Development (SCOReD)*, 2015, pp. 610–614, doi: 10.1109/SCORED.2015.7449410.

[3] J. He, Z. Wang, Z. Cui, and L. Li, "Towards an optimal trade-off of Viterbi decoder design," in *2009 IEEE International Symposium on Circuits and Systems*, 2009, pp. 3030–3033, doi: 10.1109/ISCAS.2009.5118441.

[4] M. S. Ahmed, N. S. M. Shah, Y. Y. A. -Aboosi, and A. N. Ghaleb, "A comparative study on channel coding scheme for underwater acoustic communication," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 1, pp. 176–186, 2023, doi: 10.11591/eei.v12i1.3827.

[5] J. Omura, "On the Viterbi decoding algorithm," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 177–179, 1969, doi: 10.1109/TIT.1969.1054239.

[6] J. Heller and I. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 835–848, 1971, doi: 10.1109/TCOM.1971.1090711.

[7] Y. M. Sandesh and K. Rambabu, "Implementation of convolution encoder and Viterbi decoder for constraint length 7 and bit rate 1 / 2," *International Journal of Engineering Research and Applications*, vol. 3, no. 6, pp. 42–46, 2013.

[8] R. A. Zbaid and K. K. Abdalla, "Design of convolutional encoder and Viterbi decoder using MATLAB," *International Journal for Research in Emerging Science and Technology*, vol. 1, no. 7, pp. 10–15, 2018.

[9] L. W. Couch, M. Kulkarni, and U. S. Acharya, *Digital and analog communication systems*, vol. 8. United States: Citeseer, 2013.

[10] A. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 751–772, 1971, doi: 10.1109/TCOM.1971.1090700.

[11] N. Seshadri and C. -E. W. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Transactions on Communications*, vol. 42, no. 234, pp. 313–323, 1994, doi: 10.1109/TCOMM.1994.577040.

[12] S. Shah and V. Sinha, "Iterative decoding vs Viterbi decoding: a comparison," in *Proceedings of the 14th National Conference on Communications NCCC*, 2008, pp. 491–493.

[13] Y. Y. A. -Aboosi and A. Z. Sha'ameri, "Improved underwater signal detection using efficient time–frequency de-noising technique and pre-whitening filter," *Applied Acoustics*, vol. 123, pp. 93–106, 2017, doi: 10.1016/j.apacoust.2017.03.013.

[14] Y. Y. Mohammed, "Improved time-frequency de-noising of acoustic signals for underwater detection system," Universiti Teknologi Malaysia, 2017.

[15] A. R. Murch and R. H. T. Bates, "Colored noise generation through deterministic chaos," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 5, pp. 608–613, 1990, doi: 10.1109/31.54997.

[16] H. Zhivomirov, "A method for colored noise generation," *Romanian Journal of Acoustics and Vibration*, vol. 15, no. 1, pp. 14–19, 2018.

[17] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, "Learn codes: inventing low-latency codes via recurrent neural networks," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 207–216, 2020, doi: 10.1109/JSAIT.2020.2988577.

[18] J. Wang, Y. Chen, R. Chakraborty, and S. X. Yu, "Orthogonal convolutional neural networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11502–11512, doi: 10.1109/CVPR42600.2020.01152.

[19] S. H. Prakash and V. Balamurugan, "Design of A 16 PSK Viterbi decoder for high bit data rate decoder," in *2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 2015, pp. 271–274, doi: 10.1109/ICCICCT.2015.7475288.

[20] M. N. Abdulwahed and A. K. Ahmed, "Underwater image de-nosing using discrete wavelet transform and pre-whitening filter," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 16, no. 6, pp. 2622–2629, 2018, doi: 10.12928/telkomnika.v16i6.9236.

[21] Y. Y. A. -Aboosi and A. Z. Sha'ameri, "Improved signal de-noising in underwater acoustic noise using S-transform: a performance evaluation and comparison with the wavelet transform," *Journal of Ocean Engineering and Science*, vol. 2, no. 3, pp. 172–185, 2017, doi: 10.1016/j.joes.2017.08.003.

[22] D. Jorge, C. Barbosa, A. Affonso, F. Lobo, and E. Novo, "SNR (signal-to-noise ratio) impact on water constituent retrieval from simulated images of optically complex amazon lakes," *Remote Sensing*, vol. 9, no. 7, pp. 1–18, 2017, doi: 10.3390/rs9070644.

[23] M. Aldababsa, C. Goztepe, G. K. Kurt, and O. Kucur, "Bit error rate for NOMA network," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1188–1191, 2020, doi: 10.1109/LCOMM.2020.2981024.

[24] T. Assaf, A. J. A. -Dweik, M. S. E. Moursi, H. Zeineldin, and M. A. -Jarrah, "Exact bit error-rate analysis of two-user NOMA using QAM with arbitrary modulation orders," *IEEE Communications Letters*, vol. 24, no. 12, pp. 2705–2709, 2020, doi: 10.1109/LCOMM.2020.3020161.

[25] M. Varasteh, B. Rassouli, and B. Clerckx, "On capacity-achieving distributions for complex AWGN channels under nonlinear power constraints and their applications to SWIPT," *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6488–6508, 2020, doi: 10.1109/TIT.2020.2998464.

BIOGRAPHIES OF AUTHORS

Yasin Yousif Al-Aboosi received his BE degree in electronics and communication engineering from Baghdad University, Iraq, in 1995, and his ME degree in communication engineering from Technology University, Iraq, in 2003. He received his Ph.D degree from UTM, Malaysia, in 2017. He is currently an assistant professor at the Department of Electrical Engineering, College of Engineering, Mustansiriyah University, Iraq. His main research interests are underwater acoustic communication, signal processing, and medical image processing. He can be contacted at email: alaboosiyasin@gmail.com.

Ammar Ali Sahrab received his B.Sc. degree in Electrical Engineering (Honours) from Mustansiriyah University, College of Engineering, Baghdad, Iraq, in 1997. He obtained his M.Sc. degree in Electronics and Communications Engineering from Mustansiriyah University, College of Engineering, Baghdad, Iraq, in 2000. After that, he obtained Ph.D. degree in Electronics, Telecommunications and Information Theory from University "Politehnica" of Bucharest, The Faculty of Electronics, Telecommunications and Information Technology, Bucharest, Romania, in 2016. He currently holds the position of Associate Professor in the Department of Electrical Engineering at Mustansiriyah University. His research interests are mobile communications, IoT based on microcontroller, antennas design, and implementation. He can be contacted at email: sahrab.ammar.ali@gmail.com.

Amal Ibrahim Nasser received the B.Sc. degree in electrical engineering from the University of Baghdad, Iraq on 1994, the M.Sc. degree in Electrical Engineering/Electronic and Telecommunications, Mustansiriyah University in 2000, and the Ph.D. degree in control/control and computers, University of Technology Engineering in Baghdad, Iraq dated 2007. Currently "Lecture Dr." at the College of Engineering, Al-Mustansiriya University, Iraq, Baghdad. Interested in the field of scientific research for automatic control systems and intelligence systems and their simulation in MATLAB programs. She can be contacted at email: amalalshemmiri@uomustansiriyah.edu.iq.

Hussein A. Abdulnabi received the B.Eng. degree in electrical engineering from AL-Rasheed College Baghdad/ Iraq, in 1991, and the master's degree in Electronics and Communication Engineering Science from Al-Mustansiriyah University in 2000 and the Ph.D. degree in Communication Engineering from the University of Technology, Iraq in 2017. He is currently a Assoc. Prof. in Department of Electrical Engineering, Al-Mustansiriyah University. His current research interests include communication, antenna design. He can be contacted at email: hussein_ali682@yahoo.com.