BIT Error Rate Tester for Wireless Communication Systems

Chinchu S Ragamalika
B. Tech Student
Department of Electronics & Communication Engineering
Saintgits College Of Engineering, Pathamuttom, Kottayam

Suby Maria Sojan
B. Tech Student
Department of Electronics & Communication Engineering
Saintgits College Of Engineering, Pathamuttom, Kottayam

Criss Jose
B. Tech Student
Department of Electronics & Communication Engineering
Saintgits College Of Engineering, Pathamuttom, Kottayam

Er. Sreekala K S
Assistant Professor
Department of Electronics & Communication Engineering
Saintgits College Of Engineering, Pathamuttom, Kottayam

Abstract

Bit error rate, (BER) is an important parameter used in digital communication systems that transmit digital data from one location to another. It is a powerful methodology for the end to end testing of digital transmission systems and also indicates how effective a system is. Matlab is an ideal tool for simulating digital communications systems, because of its easy scripting language and excellent data visualization capabilities. This paper presents the Matlab simulation of Bit error rate tester for wireless communication systems.

Keywords: BER, BERT, Eb/N0, POE, SNR

I. INTRODUCTION

Any digital transmission system which transmits a series of bits over a communication channel is likely to get influenced by some errors due to various factors like noise, interference etc. We need to ensure that these errors are reduced to a minimum so as not to interfere with the working of the measurements of BER on the test system. BER is a measure of how effective the system is and indicates the number of bit errors introduced in the received message. BER Testers are used to measure this value. A bit error rate (BER) is defined as the ratio of the number of errors to the total number of bits sent. This can be directly translated into the number of errors that occur in a string of bits. If there is a good medium between the transmitter and receiver, and the signal to noise ratio is high, then the bit error rate will be very small and having no noticeable effect on the system. If noise can be detected, then the bit error rate will need to be considered. Signal to noise ratios and Eb/N0 figures are the parameters that are more associated with communication systems. The Bit error rate, BER assesses the full end to end performance of a system including transmitter, receiver and a channel between them. So, bit error rate, BER enables the actual performance of a system to be tested, rather than testing the component parts.

The drawbacks of existing systems for BER measurements are discussed in the next section. Our proposed system is introduced in the third section. The procedure we have done for the BER simulation is explained in the fourth and fifth section. And finally section six deals with our simulation results.

PROPOSED SYSTEM

The proposed bit error rate tester consists of a transmitter section, a channel and a receiver section as shown in the figure below.

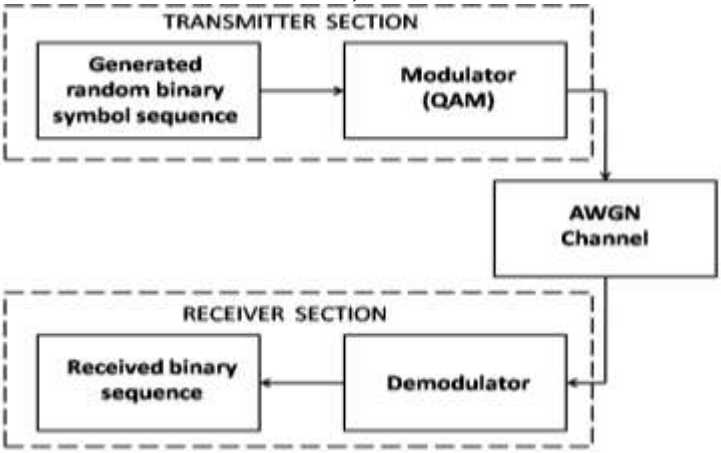

Fig. 1: The proposed BERT system

II. METHODOLOGY

MATLAB (matrix laboratory) is a calculating environment and fourth-generation programming language, developed by Math Works, and MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages. Matlab is widely used in academic and research institutions as well as industrial enterprises. It is an ideal tool for simulating digital communications systems, because of it's easy scripting language and excellent data visualization capabilities. One of the most frequent simulation tasks in the field of digital communications is the bit- error- rate testing. The bit-error-rate performance of a receiver is a figure of merit that allows different designs to be compared in a fair manner. Performing bit-error-rate testing with Matlab is very simple. In Matlab, continuous-time signals are represented with a sequence of numbers, or samples, and they are stored in a vector or an array. For communications simulations, the numeric value of the sample represents the amplitude of the continuous-time signal at a specific instant of time and assume this amplitude as a measurement of voltage.

We designed a system with a transmitter, a channel and a receiver. Then a long sequence of random bits is generated, which is the input to the transmitter. The input bits is then modulated onto some form of digital signalling, and sent though a simulated channel. Bit-error-rate performance is usually plotted on a two dimensional graph. The ordinate is the normalized signal-to-noise ratio (SNR) which is expressed as Eb/N_0: the energy-per-bit divided by the one-sided power spectral density of the noise, expressed in decibels (dB). The abscissa is the bit-error-rate, a dimensionless quantity, expressed in powers of ten. To obtain the bit-error-rate at a specific SNR, we followed the procedure given below.

1) The first step is to use the transmitter to create a digitally modulated signal from a sequence of random bits.
2) The signal-to-noise-ratio (SNR), Eb/N_0, is usually expressed in dB, and it must be converted to an ordinary ratio before we can make further use of the SNR. If we set the SNR to *m*dB, then, $Eb/N_0 = 10m/10$. So using Matlab, we find the ratio, "ebn0", from the SNR in decibels, "snrdb", as: ebn0= 10^(snrdb/10). The Eb/N_0 is a dimensionless quantity.
3) Energy per bit is the total energy of the signal, divided by the number of bits in the signal. We can express energy-per-bit as the average signal power multiplied by the duration of one bit also. The expression for *Eb* is,

$$Eb = \frac{1}{N.f_{bit}} \sum_{n=1}^{N} x^2(n)$$

where *N* is the total number of samples in the signal, and *fbit* is the bit rate in bits per second. Using Matlab, the energy per bit, "eb", of our transmitted signal, "*x*", that has a bit rate "fb", as: eb = sum(x.^2)/(length(x)*fb). Since our signal, $x(n)$, is in units of volts, the units of *Eb* are Joules.

4) Using the SNR and energy-per-bit, we can calculate N_0, the one-sided power spectral density of the noise. We have to divide *Eb* by the SNR. Using Matlab, the power spectral density of the noise, "n0", given energy- per-bit "eb", and SNR "ebn0", as: n0 = eb/ebn0. The power spectral density of the noise has units of Watts per Hertz.
5) The one-sided power spectral density of the noise, N_0, represents how much noise power is present in a 1.0 Hz bandwidth of the signal. For a real signal, $x(n)$, sampled at *fs* Hz, the noise bandwidth will be half the sampling rate. Therefore, the noise variance is

$$\sigma_n = \frac{N_0 \cdot f_s}{2}$$

where σ_n is the noise variance in W, and N_0 is the one-sided power spectral density of the noise in W/Hz. Using Matlab, the average noise power, "pn", and sampling frequency "fs", is calculated as: pn= n0*fs/2. The average noise power is in units of Watts.

6) We use one of the standard built-in functions to generate AWGN. Since the noise has a zero mean, its power and its variance are identical. We have to generate a noise vector that is having the same length as our signal vector $x(n)$, and this noise vector must have variance σn W. Using the Matlab function "randn", we generated normally distributed random numbers with a mean of zero and a variance of one. The output mus be scaled so that the result has the desired variance, σn. For that, we simply multiply the output of the "randn" function by p¾*n*. We can generate the noise vector "n", as: n = sqrt(pn)*randn(1,length(x));. The samples of the noise vector have units of volts.
7) By adding the noise vector to the signal vector we created a noisy signal. We will need to scale the resulting sum by the reciprocal of the maximum absolute value, if we are running a fixed-point simulation, so that the sum will be within the amplitude limits of ±1.0. Otherwise, we can simply add the signal vector "x" to the noise vector "n" to obtain the noisy signal vector "y" as: y = x+n;.
8) We use the receiver to demodulate the received signal. Then we compared it with the transmitted bits, to determine how many demodulated bits are in error.
9) There is an offset between the received bits and the transmitted bits, due to filtering and other delay-inducing operations of most receivers. We must first determine this offset before we can compare the two bit sequences to check for errors. Suppose our transmitted bits are stored in vector "tx", and our received bits are stored in vector "rx". Since the receiver will produce meaningless outputs while the filters are filling and flushing, the received vector should contain more bits than the transmitted vector. If the length of the transmitted bit vector is *ltx*, and the length of the received vector is *lrx*, the range of possible offsets is between zero and *lrx* –*ltx* −1. The offset is found by performing a partial cross-correlation between the two vectors. In Matlab, a partial cross-correlation, "cor", is created from bit vectors "tx" and "rx", with the following loop:

for lag= 1 : length(rx)−length(tx)−1, cor(lag)= tx*rx(lag : length(tx)−1+lag)′; end.

The resulting vector, "cor", is a partial cross-correlation of the transmitted and received bits, over the possible range of lags: 0 : *lrx –lt x* −1. We have to find the location of the maximum value of "cor", since this gives us the offset between the bit vectors. Since Matlab numbers array elements as 1 : *N* instead of as 0 :*N*−1, we have to subtract one from the correlation peak index. So using Matlab, we found the correct bit offset, "off", as: off= find(cor== max(cor))−1.

10) If we know the offset between the transmitted and received bit vectors, we can calculate the bit errors. Wherever there is a bit error, the difference between the bits will be ±1, and wherever there is not a bit error, the difference will be zero. So using Matlab, we calculated the error vector, "err", from the transmitted bit vector, "tx", and the received bit vector, "rx", having an offset of "off", as:

err = tx−rx(off+1 : length(tx)+off);.

11) Where there were bit errors the error vector, "err" contains non-zero elements in the locations. We have to tally the number of non-zero elements, since it is the total number of bit errors in the simulation. In Matlab, we calculated the total number of bit errors, "te", from the error vector "err" as:

te= sum(abs(err)).

12) Each time we simulation a bit-error-rate, we transmit and receive a fixed number of bits. Then we determine how many of the received bits are in error, then we compute the bit-error-rate as the number of bit errors divided by the total number of bits in the transmitted signal. In Matlab, the bit error-rate, "ber", is calculated as:

ber= te/length(tx),

where "te" is the total number of bit errors, and "tx" is the transmitted bit vector.

III. RESULTS AND DISCUSSION

Performing a bit-error-rate simulation is a lengthy process. We have to run individual simulations at each SNR of interest. We also need to make sure that our results are statistically significant.

A. Generated Input Random Binary Symbols:

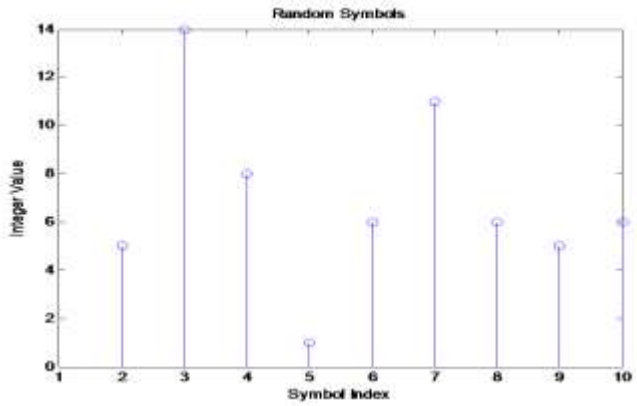

Fig. 2: Generated input random binary symbols

B. Received Signal:

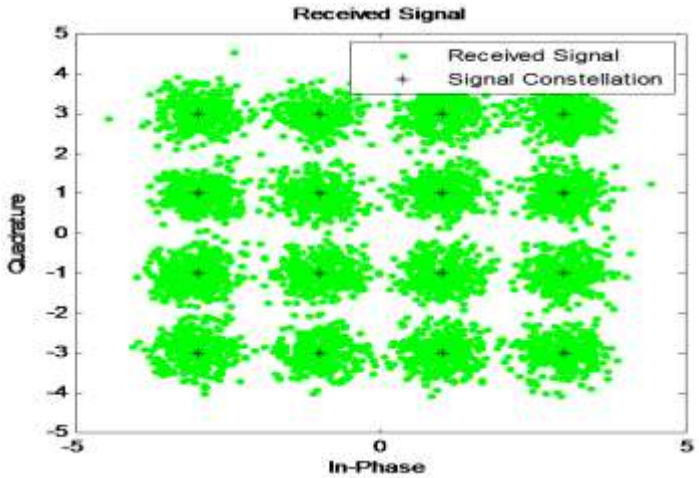

Fig. 3: Received signal BER curve

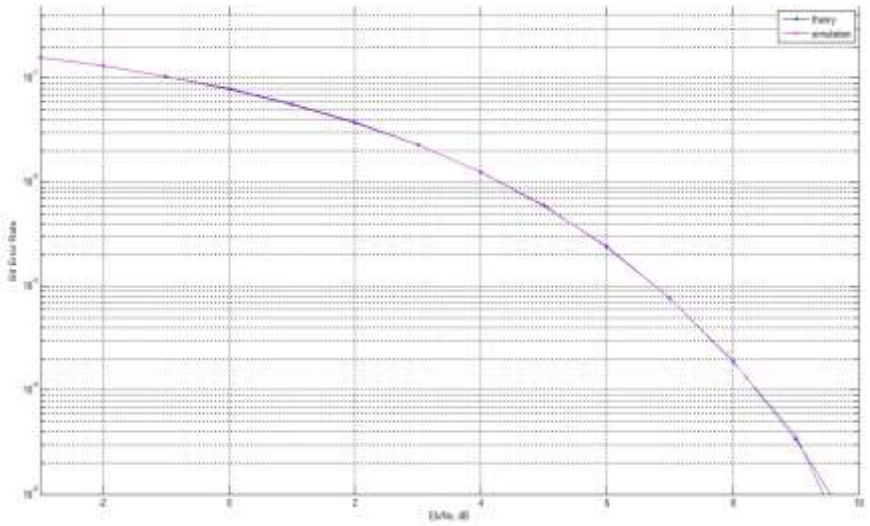

Fig. 4: BER curve

Fig. 2 represents the generated input binary symbols that is present in the transmitter side. This symbols are effectively transmitted through the channel and finally received at the receiver end. The constellation diagram at the receiver end is represented in Fig.3. The output bit error rate calculation is done by comparing it with the signal to noise ratio. This is effectively plotted in Fig.4. Since there is error in the transmitted signal, the output of the BERT is obtained as a curve as shown in Fig.4. If there is an error free transmission of symbols, then in the output of BERT, the variation of bit error rate vs Eb/N_0 is linear.

IV. CONCLUSION

A Bit-Error-Rate test is a very useful and powerful indication of the performance of a communication system that can be directly related to its operational performance. Bit error rate testing, is a powerful methodology for end to end testing of digital transmission systems. If the Bit-Error-Rate rises high then the system performance will degrade. The system will operate satisfactorily if it doesn't. We simulated the Bit-error-rate performance by adding a controlled amount of noise to the transmitted signal. This noisy signal then becomes the input to the receiver. Then the receiver demodulates the signal, producing a recovered bits sequence. Finally, we compared the received bits with transmitted bits, and plotted the Bit-Error-Rate versus Eb/N_0 in both practical theoretical cases.

ACKNOWLEDGMENT

First of all, we thank God Almighty for His abundant grace and mercy which enabled me in the finalization of this project. The support and help of a few people not only enabled us to complete our project successfully, but also made it a worthwhile experience. We thank the management for providing all the infrastructure and facilities. We are grateful to Dr.M.C.Philipose, Principal, SAINTGITS College of Engineering for providing the best facilities and atmosphere for doing this project. We thank Prof. Susan Abe, HOD, Electronics and Communication, for her encouragement and support. Also we thank our project guide Er. Sreekala. K. S, Asst. Prof. Dept. of Electronics and Communication for her valuable suggestions, ideas and support without which this project would have been a dream. Finally, we thank all the staff members of our department for their timely help and guidance.

REFERENCES

[1] Breed, G. (2003, January). Bit ErrorRate: Fundamental concepts and measurement issues. Retrieved April 07, 2014, from high frequency electronics: http://www.highfrequencyelectronics.com/Archives/Jan03/HFE0103_Tutorial.pdf[
[2] Bohdanowicz, A. (n.d.). On Efficient BER Evaluation of Digital Communication Systems via Importance Sampling. RetrievedApril7,2014,fromCiteSeerx: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.2785&rep=rep1&type=pdf
[3] Peter J. Smith, Mansoor Sha_ and Hongsheng Gao, \Quick sim-ulation: A review of importance sampling techniques in communication systems," IEEE Journal on Selected Areas in Communications, vol. 15, pp. 597{613, May 1997.
[4] A. Alimohammad, S. F. Fard, and B. F. Cockburn, "FPGA-accelerated baseband design and verification of broadband MIMO wireless systems,"in Proc. IEEE 1st Int. Conf. Adv. Syst. Testing Validation, Sep. 2009,pp. 135–140.
[5] A. Alimohammad, S. F. Fard, and B. F. Cockburn, "FPGA-based accelerator for the verification of leading-edge wireless systems," in Proc.IEEE Int DAC,Jul 2009,pp 844-847.
[6] Lakshmy Sukumaran, D. K. (2014). Bit Error Rate Testers - A Study. International Journal For Engineering Research and Applications, ISSN : 2248-9622, Vol. 4, Issue 4(Version 8), April 2014, pp.122-125.
[7] http://www.math.utah.edu/~wright/misc/matlab/matlabintro.html
[8] WIKIPEDIA, FREE ENCYCLOPAEDIA, ARTICLE ON BIT ERROR RATE HTTP://EN.WIKIPEDIA.ORG/WIKI/BIT_ERROR_RATE.
[9] WIKIPEDIA, FREE ENCYCLOPAEDIA, ARTICLE ON SIGNAL TO NOISE RATIO HTTP://EN.WIKIPEDIA.ORG/WIKI/S/N_RATIO