Volume 4, Issue 6, June 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Simulation of Bit Error Rate with the help of Matlab

Monika Sharma
M.Tech Student
Department of ECE
Eternal University, Baru Sahib
Himachal Pradesh-173101, India

ABSTRACT- Bit error rate is used in accessing systems that transfer digital data from one place to another. It is the rate with which errors occur in a transmission system. Bit error rate is applicable in various systems like fiber optic data systems. For the simulation of digital communication system, Matlab is an ideal tool. It is refer as ideal because of its easy scripting language and amazing data visualization capabilities. We can easily perform bit error rate testing using Matlab but it does require some prior knowledge. Different designs can be compared by using bit error rate performance of a receiver in an absolutely fair manner. This paper will discuss simulation of bit error rate using a tool Matlab.

KEYWORDS- Matlab, Bit-Error- Rate

I. INTRODUCTION

In digital transmission, the number of bit errors is the number of received bits of a data stream over a communication channel that have been altered due to noise, interference, distortion or bit synchronization errors. The bit error rate or bit error ratio (BER) is the number of bit errors divided by the total number of transferred bits during a studied time interval. BER is a unit less performance measure, often expressed as a percentage. The bit error probability p_e is the expectation value of the BER. The BER can be considered as an approximate estimate of the bit error probability. This estimate is accurate for a long time interval and a high number of bit errors.
BER = number of errors / total number of bits sent

Example:
As an example, assume this transmitted bit sequence:
0 1 1 0 0 0 1 0 1 1,
and the following received bit sequence:
0 0 1 0 1 0 1 0 0 1,
The number of bit errors (the underlined bits) is in this case 3. The BER is 3 incorrect bits divided by 10 transferred bits, resulting in a BER of 0.3 or 30%.
The bit error rate, BER, can also be defined in terms of the probability of error or POE. The determine this, three other variables are used. They are the error function, erf, the energy in one bit, Eb, and the noise power spectral density (which is the noise power in a 1 Hz bandwidth), No. It should be noted that each different type of modulation has its own value for the error function. This is because each type of modulation performs differently in the presence of noise. In particular, higher order modulation schemes (e.g. 64QAM, etc) that are able to carry higher data rates are not as robust in the presence of noise. Lower order modulation formats (e.g. BPSK, QPSK, etc.) offer lower data rates but are more robust. The energy per bit, Eb, can be determined by dividing the carrier power by the bit rate and is a measure of energy with the dimensions of Joules. No is a power per Hertz and therefore this has the dimensions of power (joules per second) divided by seconds). Looking at the dimensions of the ratio Eb/No all the dimensions cancel out to give a dimensionless ratio.

II. FACTOR AFFECTING BIT ERROR RATE

BER may be affected by transmission channel noise, interference, distortion, bit synchronization problems, attenuation, wireless multipath fading, etc. The BER may be improved by choosing a strong signal strength (unless this causes cross-talk and more bit errors), by choosing a slow and robust modulation scheme or line scheme, and by applying channel coding schemes such as redundant forward error correction codes.
The transmission BER is the number of detected bits that are incorrect before error correction, divided by the total number of transferred bits (including redundant error codes). The information BER, approximately equal to the decoding error probability, is the number of decoded bits that remain incorrect after the error correction, divided by the total number of decoded bits (the useful information). Normally the transmission BER is larger than the information BER. The information

BER is affected by the strength of the forward error correction code. The interference levels present in a system are generally set by external factors and cannot be changed by the system design. However it is possible to set the bandwidth of the system. By reducing the bandwidth the level of interference can be reduced. However reducing the bandwidth limits the data throughput that can be achieved. It is also possible to increase the power level of the system so that the power per bit is increased. Lower order modulation schemes can be used, but this is at the expense of data throughput. It is necessary to balance all the available factors to achieve a satisfactory bit error rate.

III. SIMULATION TOOL (MATLAB)

MATLAB is widely used in all areas of applied mathematics, in education and research at universities, and in the industry. MATLAB stands for MATrix LABoratory and the software is built up around vectors and matrices. This makes the software particularly useful for linear algebra but MATLAB is also a great tool for solving algebraic and differential equations and for numerical integration. MATLAB has powerful graphic tools and can produce nice pictures in both 2D and 3D. It is also a programming language, and is one of the easiest programming languages for writing mathematical programs. MATLAB also has some tool boxes useful for signal processing, image processing, optimization, etc.

In Matlab, we represent continuous-time signals with a sequence of numbers, or samples, which are generally stored in a vector or an array. Before we can performance bit-error-rate test, we must precisely understand the meaning of these samples. We must know what aspect of the signal the value of these samples represents. We must also know the time interval between successive samples. For communications simulations, the numeric value of the sample represents the amplitude of the continuous-time signal at a specific instant in time.

IV. PROCEDURE FOR SIMULATION

1. Run Transmitter

The first step in the simulation is to use the transmitter to create a digitally modulated signal from a sequence of pseudo-random bits. Once we have created this signal, $x(n)$, we need to make some measurements of it.

2. Establish SNR

The signal-to-noise-ratio (SNR), E_b/N_0, is usually expressed in decibels, but we must convert decibels to an ordinary ratio before we can make further use of the SNR. If we set the SNR to m dB, then $E_b/N_0 = 10m/10$.

Using Matlab, we find the ratio, 'ebn0', from the SNR in decibels, 'snrdb', as:

ebn_0= 10^(snrdb/10).

Note that E_b/N_0 is a dimensionless quantity.

3. Determine *Eb*

Energy-per-bit is the total energy of the signal, divided by the number of bits contained in the signal. We can also express energy-per-bit as the average signal power multiplied by the duration of one bit. Either way, the expression for *Eb* is:

$$E_b = \frac{1}{N \cdot f_{bit}} \sum_{n=1}^{N} x^2(n),$$

where N is the total number of samples in the signal, and *fbi t* is the bit rate in bits-per-second.

Using Matlab, we find the energy-per-bit, 'e_b', of our transmitted signal, 'x', that has a bit rate 'fb', as:

e_b = sum(x.^2)/(length(x)∗fb).

Since our signal, $x(n)$, is in units of volts, the units of E_b are Joules.

4. Calculate N_0

With the SNR and energy-per-bit now known, we are ready to calculate $N0$, the one- sided power spectral density of the noise. All we have to do is divide E_b by the SNR, providing we have converted the SNR from decibels to a ratio.

Using Matlab, we find the power spectral density of the noise, 'n_0', given energy-per-bit 'e_b', and SNR 'ebn_0', as:

n_0 = eb/ebn_0.

The power spectral density of the noise has units of Watts per Hertz.

5. Calculate σ_n

The one-sided pc spectral density of the noise, $N0$, tells us how much noise power is present in a 1.0 Hz bandwidth of the signal. In order to find the variance, or average power, of the noise, we must know the noise bandwidth.

For a real signal, $x(n)$, sampled at fs Hz, the noise bandwidth will be half the sampling rate. Therefore, we find the average power of the noise by multiplying the power spectral density of the noise by the noise bandwidth:

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

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', of noise having power spectral density 'n0', and sampling frequency 'fs', is calculated as:

pn= n0∗fs/2.

The average noise power is in units of Watts.

6. Generate Noise

Although the communications toolbox of Matlab has functions to generate additive white Gaussian noise, we will 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 need to generate a noise vector that is the same length as our signal vector *x*(*n*), and this noise vector must have variance W. The Matlab function 'randn' generates normally distributed random numbers with a mean of zero and a variance of one. We must scale the output so the result has the desired variance, .To do this, 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));.

Like the signal vector, the samples of the noise vector σ_n have units of volts.

7. Add Noise

We create a noisy signal by adding the noise vector to the signal vector. If we are running a fixed-point simulation, we will need to scale the resulting sum by the reciprocal of the maximum absolute value, so the sum stays within amplitude limits of ±1.0. Otherwise, we can simply add the σ_n ıl vector 'x' to the noise vector 'n' to obtain the noisy signal vector 'y' as: y = x+n;.

8. Run Receiver

Once we have created a noisy signal vector, we use the receiver to demodulate this signal. The receiver will produce a sequence of demodulated bits, which we must compare to the transmitted bits, in order to determine how many demodulated bits are in error.

9. Determine Offset

Due to filtering and other delay-inducing operations typical of most receivers, there will be an offset between the received bits and the transmitted bits. Before we can compare the two bit sequences to check for errors, we must first determine this offset. One way to do this is by correlating the two sequences, then searching for the correlation peak. Suppose our transmitted bits are stored in vector 'tx', and our received bits are stored in vector 'rx'. The received vector should contain more bits than the transmitted vector, since the receiver will produce (meaningless) outputs while the filters are filling and flushing. If the length of the transmitted bit vector is *lt x* , and the length of the received vector is *lrx* , the range of possible offsets is between zero and *lrx* −*lt x* −1. We can find the offset by performing a partial cross-correlation between the two vectors.

Using Matlab, we can create a partial cross-correlation, 'cor', 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 need to find the location of the maximum value of 'cor', since this will tell us the offset between the bit vectors. Since Matlab numbers array elements as 1 : *N* instead of as 0 :*N*−1, we need to subtract one from the index of the correlation peak. Using Matlab, we find the correct bit offset, 'off', as:

off= find(cor== max(cor))−1.

10. Create Error Vector

Once we know the offset between the transmitted and received bit vectors, we are ready to calculate the bit errors. For bit values of zero and one, a simple difference will reveal 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.

Using Matlab, we calculate 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. Count Bit Errors

The error vector, 'err' contains non-zero elements in the locations where there were bit errors. We need to tally the number of non-zero elements, since this is the total number of bit errors in this simulation.

Using Matlab, we calculate the total number of bit errors, 'te', from the error vector 'err' as:
te= sum(abs(err)).

12. Calculate Bit-Error-Rate

Each time we run a bit-error-rate simulation; we transmit and receive a fixed number of bits. We determine how many of the received bits are in error, then compute the bit-error-rate as the number of bit errors divided by the total number of bits in the transmitted signal.
Using Matlab, we compute the bit-error-rate, 'ber', as:
ber= te/length(tx), where 'te' is the total number of bit errors, and 'tx' is the transmitted bit vector.

V. SIMULATION RESULTS

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

1. Statistical Validity

When the bit-error-rate is high, many bits will be in error. The worst-case bit-error-rate is 50 percent, at which point, the modem is essentially useless. Most communications systems require bit-error-rates several orders of magnitude lower than this.
Even a bit-error-rate of one percent is considered quite high.
We usually want to plot a curve of the bit-error-rate as a function of the SNR, and include enough points to cover a wide range of bit-error-rates. At high SNRs, this can become difficult, since the bit-error-rate becomes very low. For example, a bit-error-rate of 10−6 means only one bit out of every million bits will be in error.
If our test signal only contains 1000 bits, we will most likely not see an error at this bit-error-rate.
In order to be statistically significant, each simulation we run must generate some number of errors. If a simulation generates no errors, it does not mean the bit-error-rate is zero; it only means we did not have enough bits in our transmitted signal. As a rule of thumb, we need about 100 (or more) errors in each simulation, in order to have confidence that our bit-error-rate is statistically valid. At high SNRs, this can require a test signal containing millions, or even billions of bits.

2. Plotting

Once we perform enough simulations to obtain valid results at all SNRs of interest, we will plot the results. We begin by creating vectors for both axes. The X-axis vector will contain SNR values, while the Y-axis vector will contain bit-error-rates. The Y- axis should be plotted on a logarithmic scale, whereas the X-axis should be plotted on a linear scale.
Fig. 1, 2 and 3 shows example of plot of the results of a bit-error-rate simulation at different E_b/N_0.

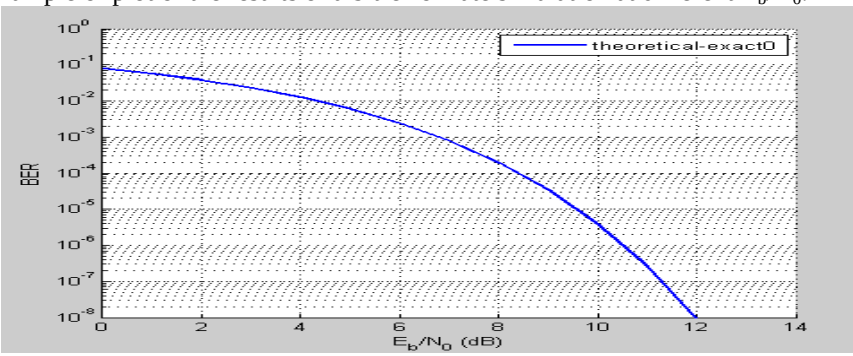

Fig-1

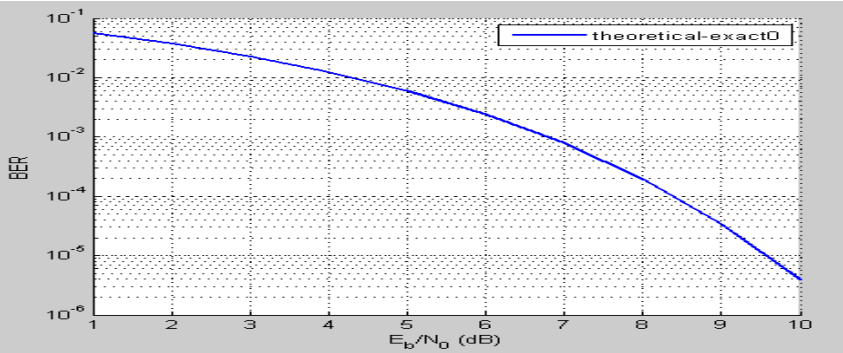

Fig-2

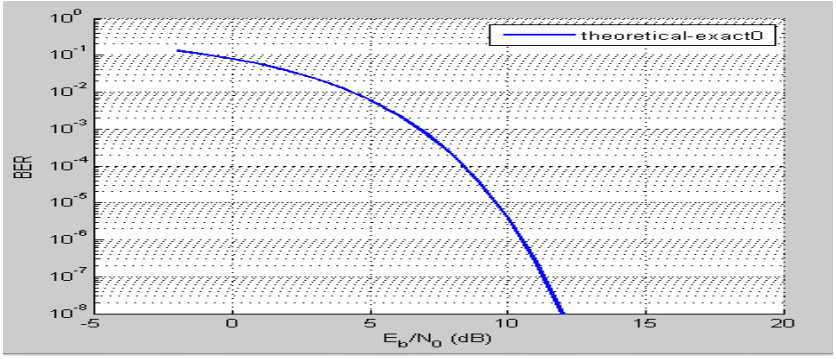

Fig-3

VI. CONCLUSION

Bit error rate displays an excellent indication of the performance of a data link such as radio or fiber optic method. The key parameters of importance in any data link is the number of errors that occur, the bit error rate is a main parameter. Bit-Error-Rate also enables other features of the link such as the power, bandwidth, etc. to be altered to enable the required performance to be obtained. A Bit-Error-Rate test provides a very useful indication of the performance of 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. If it doesn't then the system will operate satisfactorily. We simulate the Bit-error-rate performance by adding a controlled amount of noise to the transmitted signal. This signal with noise then becomes the receiver's input. Then the receiver demodulates the signal, producing a recovered bits sequence. Finally, the received bits are compared with transmitted bits, and match up the errors through Bit-Error-Rate versus Eb/N0 plot.

REFERENCES

[1] http://www.math.utah.edu/~wright/misc/matlab/matlabintro.html
[2] WIKIPEDIA, FREE ENCYCLOPAEDIA, ARTICLE ON BIT ERROR RATE HTTP://EN.WIKIPEDIA.ORG/WIKI/BIT_ERROR_RATE.
[3] WIKIPEDIA, FREE ENCYCLOPAEDIA, ARTICLE ON SIGNAL TO NOISE RATIO HTTP://EN.WIKIPEDIA.ORG/WIKI/S/N_RATIO
[4] M. JERUCHIM, (1994), "TECHNIQUES FOR ESTIMATING THE BIT ERROR RATE IN THE SIMULATION OF DIGITAL COMMUNICATION SYSTEMS", IEEE J. SELECT. AREAS COMMUNICATION., VOL. SAC-2, PP. 153-170, JAN.1994.
[5] THE MATH WORKS, INC., THE STUDENT EDITION OF MATLAB VERSION 7 USER'S GUIDE, PRENTICE HALL, ISBN 0-13-184979-4, 1995.
[6] JOHN G. PROAKIS and MASOUD SALEHI, "COMMUNICATION SYSTEM USING MATLAB" THOMSON ASIA PVT. LTD., SINGAPORE, 2003.
[7] Yu Tang, Xiao-lan Lv "Research on the modulation and demodulation of BPSK and BDPSK simulator based on Matlab",IEEE transactions ,2011
[8] http://conference.ieee passau.org/fileadmin/templateConf2012/images/papers/10_Bit_Error_Rate_Testing_Serial_Communication_Equipment_using_PseudoRandom_Bit_Sequences.pdf
[9] http://www.ijera.com/papers/Vol3_issue1/DB31706711.pdf.

