American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-6, Issue-4, pp-52-59
www.ajer.org
Research Paper Open Access

End To End Testing of Digital Transmission Systems Using Matlab

Dr. Ahlal H Montaser Mohamed[1], Dr. Amamer Khalil Masoud Ahmidat[2]
[1]*Faculty of IT University of Tripoli, Libya*
[2]*Faculty Of Science. Azzaytuna University, Baniwalid, Libya*

ABSTRACT: *Bit error rate, (BER) is a key parameter that is used in assessing systems that transmit digital data from one location to another. Systems for which bit error rate, is applicable include radio data links as well as fiber optic data systems, Ethernet, or any system that transmits data over a network of some form where noise, interference, and phase jitter may cause quality degradation of the digital signal. MATLAB has an ideal tool for simulating digital communications systems over the network. One of the most frequent simulation tasks in the field of digital communications is bit-error-rate testing of network. The bit-error-rate performance of a receiver is a figure of merit that allows different designs to be compared in a fair manner. BER Tool is an interactive GUI for analyzing communication systems' bit error rate (BER) performance.*

Keywords: *E_b/N_o range*

I. INTRODUCTION

BER is directly translated into the number of errors that occur in a string of a stated number of bits. The definition of bit error rate can be translated into a simple formula:

BER = number of errors / total number of bits sent

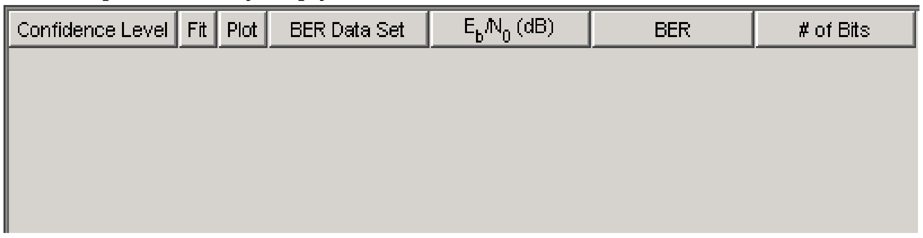
Using BERTool it can

- Generate BER data for a communication system using
  - Closed-form expressions for theoretical BER performance of selected types of communication systems.
  - The semi analytic technique.
  - Simulations contained in MATLAB simulation functions or Simulink® models. After it create a function or model that simulates the system, BERTool iterates over choice of E_b/N_0 values and collects the results.
- Plot one or more BER data sets on a single set of axes. It can graphically compare simulation data with theoretical results or simulation data from a series of similar models of a communication system.
- Fit a curve to a set of simulation data. Send BER data to the MATLAB workspace or to a file for a processing it might want to perform.

If the medium between the transmitter and receiver is good and the signal to noise ratio is high, then the bit error rate will be very small possibly insignificant and having no noticeable effect on the overall system However if noise is detected, then there is chance that the bit error rate will need to be considered. Although there are some differences in the way these systems work and the way in which bit error rate is affected, the basics of bit error rate itself are still the same. When data is transmitted over a data link, there is a possibility of errors being introduced into the system. If errors are introduced into the data, then the integrity of the system may be compromised. As a result, it is necessary to assess the performance of the system, and bit error rate, BER, provides an ideal way in which this can be achieved. Unlike many other forms of assessment, bit error rate, BER assesses the full end to end performance of a system including the transmitter, receiver and the medium between the two. In this way, bit error rate, BER enables the actual performance of a system in operation to be tested, rather than testing the component parts and hoping that they will operate satisfactorily when in place. The main reasons for the degradation of a data channel and the corresponding bit error rate, BER is noise and changes to the propagation path (where radio signal paths are used). Both effects have a random element to them, the noise following a Gaussian probability function while the propagation model follows a Rayleigh model. This means that analysis of the channel characteristics are normally undertaken using statistical analysis techniques. For fiber optic systems, bit errors mainly result from imperfections in the components used to make the link. These include the optical driver, receiver, connectors and the fiber itself. Bit errors may also be introduced as a result of optical dispersion and attenuation that may be present. Also noise may be introduced in the optical receiver itself. Another contributory factor for bit errors is any phase jitter that may be present in the system as this can alter the sampling of the data. Signal to noise ratios and Eb/No figures are parameters that are more associated with radio links and radio communications systems. In terms of this, 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 that are able to carry higher data rates are not as robust in the presence of noise. Lower order modulation formats offer lower data rates but is more robust. Looking at the dimensions of the ratio Eb/No all the dimensions cancel out to give a dimensionless ratio. It is important to note that POE is proportional to Eb/No and is a form of signal to noise ratio.

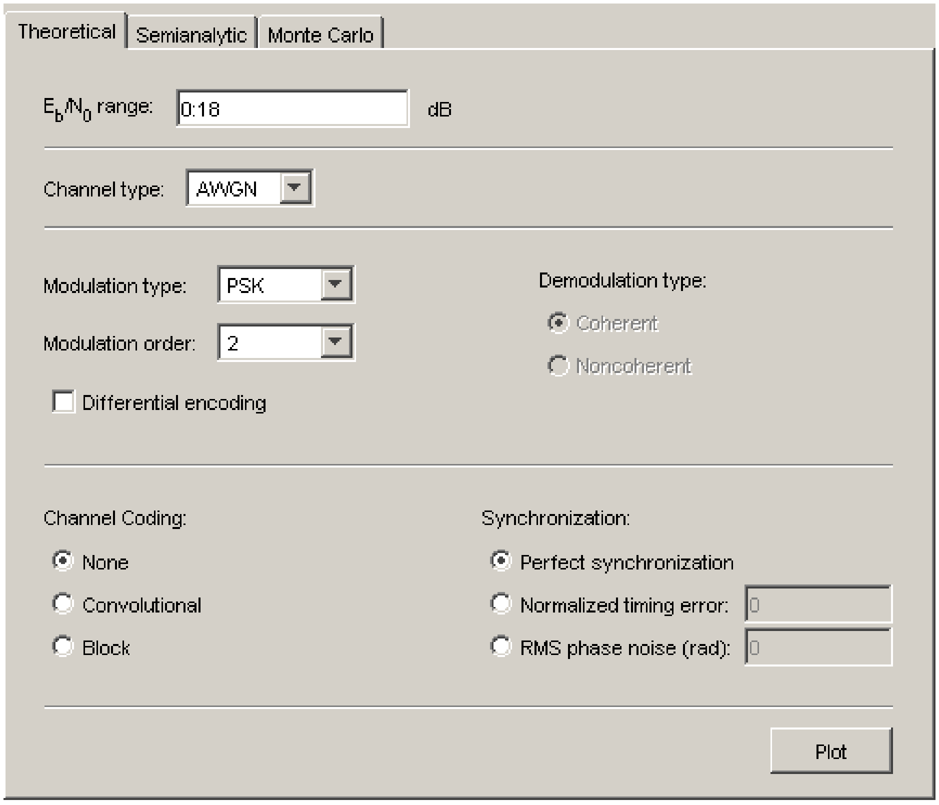
II. COMPONENT OF BER TOOL

A data viewer at the top. It is initially empty shown as below.



Confidence Level	Fit	Plot	BER Data Set	E_b/N_0 (dB)	BER	# of Bits

After it instruct BERTool to generate one or more BER data sets, they appear in the data viewer. A set of tabs on the bottom. Labeled Theoretical, Semianalytic, and Monte Carlo, the tabs correspond to the different methods by which BERTool can generate BER data.



Theoretical | Semianalytic | Monte Carlo

E_b/N_0 range: 0:18 dB

Channel type: AWGN

Modulation type: PSK

Modulation order: 2

☐ Differential encoding

Demodulation type:
[✓ Coherent]
Noncoherent

Channel Coding:
[✓ None]
Convolutional
Block

Synchronization:
[✓ Perfect synchronization]
Normalized timing error: 0
RMS phase noise (rad): 0

The components of BERTool act as one integrated tool. These behaviors reflect their integration:

- If select a data set in the data viewer, BERTool reconfigures the tabs to reflect the parameters associated with that data set and also highlights the corresponding data in the BER Figure window. This is useful if the data viewer displays multiple data sets and it want to recall the meaning and origin of each data set.

- If click data plotted in the BER Figure window, BERTool reconfigures the tabs to reflect the parameters associated with that data and also highlights the corresponding data set in the data viewer.
- If configure the Semi analytic or Theoretical tab in a way that is already reflected in an existing data set, BERTool highlights that data set in the data viewer. This prevents BERTool from duplicating its computations and its entries in the data viewer, while still showing the results that it requested.
- If it select options in the data viewer that affect the BER plot, the BER Figure window reflects selections immediately. Such options relate to data set names, confidence intervals, curve fitting, and the presence or absence of specific data sets in the BER plot.

III. SIMULATION PROCEDURE

Bit-error-rate testing requires a transmitter, a receiver, and a channel. It begin by generating a long sequence of random bits, which it provide as input to the transmitter. The transmitter modulates these bits onto some form of digital signaling, which it will send though a simulated channel. Bit-error-rate performance is usually depicted on a two dimensional graph. The ordinate is the normalized signal-to-noise ratio (SNR) expressed as E_b/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, usually expressed in powers of ten. To create a graph of bit-error-rate versus SNR, it plot a series of points. Each of these points requires us to run a simulation at a specific value of SNR. To obtain the bit-error-rate at a specific SNR, it follow the procedure given below

A. 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 it has created this signal, $x(n)$, it need to make some measurements of it.

B. Establish SNR

The signal-to-noise-ratio (SNR), Eb/N0, is usually expressed in decibels, but it must convert decibels to an ordinary ratio before it can make further use of the SNR. If it set the SNR to m dB, then Eb/N0= 10m/10. Using MATLAB, it find the ratio, ebn0, from the SNR in decibels, snrdb, as: ebn0= 10^(snrdb/10). Note that Eb/N0 is a dimensionless quantity.

C. Determine E_b

Energy-per-bit is the total energy of the signal, divided by the number of bits contained in the signal. It 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: *Eb*= where *N* is the total number of samples in the signal, and *fbit*is the bit rate in bits-per-second. Using MATLAB, it find 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.

D. Calculate N_0

With the SNR and energy-per-bit now known, it is ready to calculatedN_0, the one-sided power spectral density of the noise. All have to do is divide $E_b$by the SNR, providing it has converted the SNR from decibels to a ratio. Using MATLAB, it find the power spectral density of the noise, N_0, 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.

E. Calculate □$_n$

The one-sided power spectral density of the noise, N0, tells 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, it 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, it find the average power of the noise by multiplying the power spectral density of the noise by the noise bandwidth, where σn is the noise variance in W, and N_0 is the one-sided power spectral density of the noise inW/Hz.

F. Generate Noise

Although the communications toolbox of MATLAB has functions to generate additive white Gaussian noise, it 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. It need to generate a noise vector that is the same length as our signal vector x(n), and this noise vector must have variance σn W. The MATLAB function „randn generates normally distributed random numbers with a mean of zero and a variance of one. It must scale the output so the result has the desired variance, σn. To do this, it simply multiply the output of the „randn function by p¾n. It can generate the noise vector „n, as:
n = sqrt(pn)*randn(1,length(x));.

G. Add Noise

It create a noisy signal by adding the noise vector to the signal vector. If it is running a fixed-point simulation, it 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, it can simply add the signal vector x to the noise vector n to obtain the noisy signal vector y as: y = x + n.

H. Run Receiver

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

I. 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 it can compare the two bit sequences to check for errors, it 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 −1. It can find the offset by performing a partial cross-correlation between the two vectors. Using MATLAB, it 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 : −1. It 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, it need to subtract one from the index of the correlation peak.

J. Create Error Vector

Once it know the offset between the transmitted and received bit vectors, it is 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, it 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);.

K. Count Bit Errors

The error vector, err contains non-zero elements in the locations where there were bit errors. It needs to tally the number of non-zero elements, since this is the total number of bit errors in this simulation. Using MATLAB, it calculate the total number of bit errors, te, from the error vector err as: te= sum(abs(err)).

L. Calculate Bit-Error-Rate

Each time it run a bit-error-rate simulation, it transmit and receive a fixed number of bits. It 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, it 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.

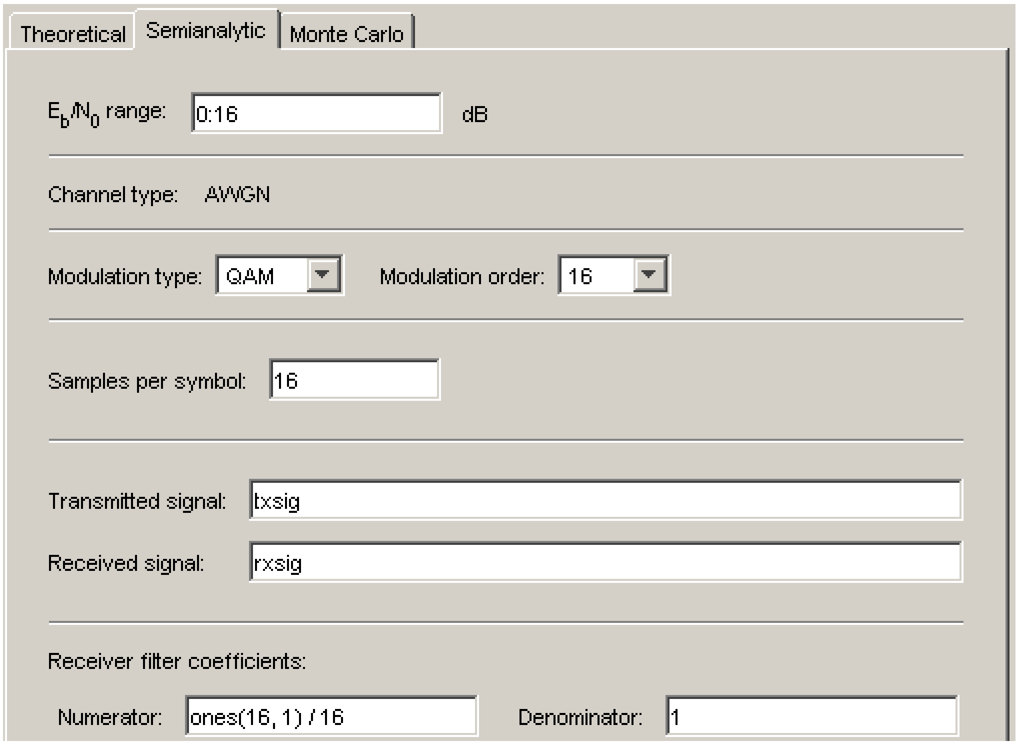
IV. USING SEMIANALYTIC TECHNIQUE TO COMPUTE BER

To set up the transmitted and received signals, run code:

```
% Step 1. Generate message signal of length >= M^L.
M = 16; % Alphabet size of modulation
L = 1; % Length of impulse response of channel
msg = [0:M-1 0]; % M-ary message sequence of length > M^L

% Step 2. Modulate the message signal using baseband modulation.
modsig = qammod(msg,M); % Use 16-QAM.
Nsamp = 16;
```

```
modsig = rectpulse(modsig,Nsamp); % Use rectangular pulse shaping.

% Step 3. Apply a transmit filter.
txsig = modsig; % No filter in this example

% Step 4. Run txsig through a noiseless channel.
rxsig = txsig*exp(1i*pi/180); % Static phase offset of 1 degree
```

Open BERTool and go to the Semi analytic tab. Set parameters as shown in the following figure and Click Plot.



E_b/N_0 range: 0:16 dB

Channel type: AWGN

Modulation type: QAM Modulation order: 16

Samples per symbol: 16

Transmitted signal: txsig

Received signal: rxsig

Receiver filter coefficients:

Numerator: ones(16, 1) / 16 Denominator: 1

Procedure for Using the Semi analytic Tab in BERTool

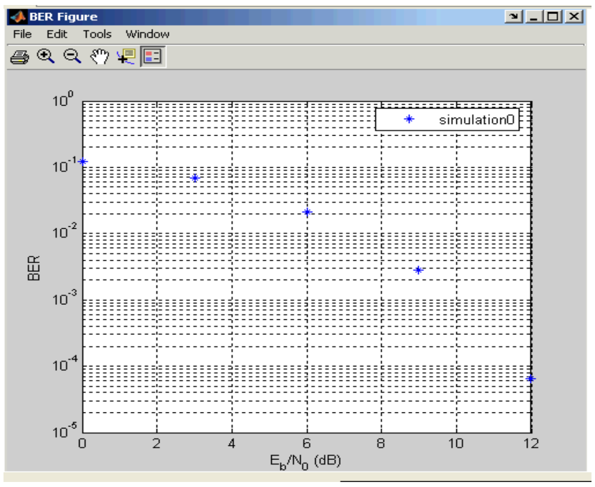
The procedure below describes how it typically implement the semi analytic technique using BERTool:

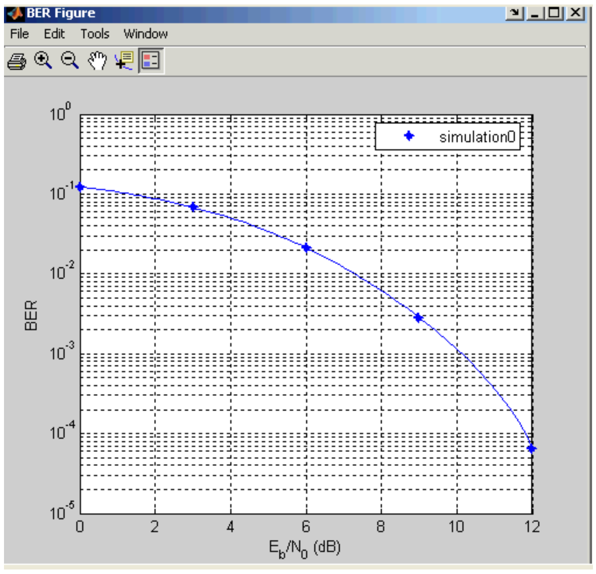
1. Generate a message signal containing *at least* M^L symbols, where M is the alphabet size of the modulation and L is the length of the impulse response of the channel in symbols. A common approach is to start with an augmented binary pseudo noise (PN) sequence of total length $(\log_2 M)M^L$. An *augmented* PN sequence is a PN sequence with an extra zero appended, which makes the distribution of ones and zeros equal.
2. Modulate a carrier with the message signal using baseband modulation. Supported modulation types are listed on the reference page for semi analytic. Shape the resultant signal with rectangular pulse shaping, using the oversampling factor that it will later use to filter the modulated signal. Store the result of this step as taxis for later use.
3. Filter the modulated signal with a transmit filter. This filter is often a square-root raised cosine filter, but it can also use a Butterworth, Bessel, Chebyshev type 1 or 2, elliptic, or more general FIR or IIR filter. If use a square-root raised cosine filter, use it on the non over sampled modulated signal and specify theoversampling factor in the filtering function. If it use another filter type, it can apply it to the rectangular pulse shaped signal.
4. Run the filtered signal through a *noiseless* channel. This channel can include multipath fading effects, phase shifts, amplifier nonlinearities, quantization, and additional filtering, but it must not include noise. Store the result of this step as rxsig for later use.

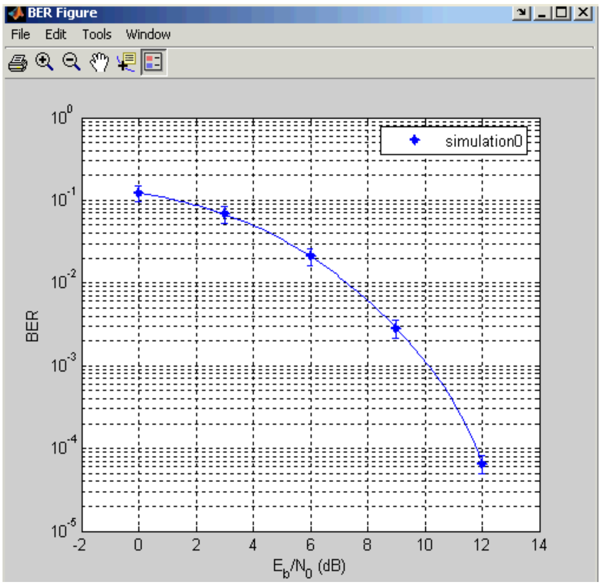
V. SIMULATION RESULTS

Performing a bit-error-rate simulation can be a lengthy process. Itneed to run individual simulations at each SNR of interest. It also need to make sure our results are statistically significant. 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. It 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, it will most likely not see an error at this Bit-error-rate. In order to be statistically significant, each simulation it 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 it did not have enough bits in our transmitted signal. As a rule of thumb, it 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.

BER Tool loads the model into memory (which in turn initializes several variables in the MATLAB workspace), runs the simulation once for each value of Eb/N0, and gathers BER data. BERTool creates a listing in the data viewer and BER Tool plots the data in the BER Figure window:







Pseudo Code for BER simulation

```
function [ber, numBits] = bertooltemplate(EbNo, maxNumErrs, maxNumBits)
% Import Java class for BERTool.
importcom.mathworks.toolbox.comm.BERTool;
% Initialize variables related to exit criteria.
totErr = 0;  % Number of errors observed
numBits = 0; % Number of bits processed
% --- Set up parameters. ---
% --- INSERT CODE HERE.
% Simulate until number of errors exceeds maxNumErrs
% or number of bits processed exceeds maxNumBits.
while((totErr<maxNumErrs) && (numBits<maxNumBits))

  % Check if the user clicked the Stop button of BERTool.
if (BERTool.getSimulationStop)
break;
end
% --- Proceed with simulation.
  % --- Be sure to update totter and numBits.
% --- INSERT CODE HERE.
end % End of loop
% Compute the BER.
beer = totter/numBits;
% Set up initial parameters.
siglen = 1000; % Number of bits in each trial
M = 2; % DBPSK is binary.
hMod = modem.dpskmod('M', M); % Create a DPSK modulator
hDemod = modem.dpskdemod(hMod); % Create a DPSK
                % demodulator using the modulator object
snr = EbNo; % Because of binary modulation
ntrials = 0; % Number of passes through the loop
msg = randint(siglen, 1, M); % Generate message sequence.
txsig = modulate(hMod, msg); % Modulate.
rxsig = awgn(txsig, snr, 'measured'); % Add noise.
decodmsg = demodulate(hDemod, rxsig); % Demodulate.
newerrs = biterr(msg,decodmsg); % Errors in this trial
ntrials = ntrials + 1; % Update trial index.
% Update the total number of errors.
totter = totter + newerrs;
% Update the total number of bits processed.
```

numBits = ntrials * siglen;

VI. CONCLUSION

Bit error rate BER is a parameter which gives an excellent indication of the performance of a data link such as radio or fiber optic system. As one of the main parameters of interest in any data link is the number of errors that occur, the bit error rate is a key parameter. Knowledge of the BER also enables other features of the link such as the power and bandwidth, etc to be tailored to enable the required performance to be obtained. Bit error rate (BER) testing, is a powerful methodology for end to end testing of digital transmission systems. A BER test provides a measurable and useful indication of the performance of the performance of the system that can be directly related to its operational performance. If the BER rises too high then the system performance will noticeably degrade. If it is within limits then the system will operate satisfactorily. It simulate the Bit-error-rate performance of digital communication system by adding a controlled amount of noise to the transmitted signal. This noisy signal then becomes the input to the receiver. The receiver demodulates the signal, producing a sequence of recovered bits. Finally, it compare the received bits to the transmitted bits, and tally up the errors through BER versus E_b/N_0 plot.

REFERENCES

[1]. D. Hansel Man And B. Littlefield, Mastering Matlab 7. A Comprehensive Tutorial And Reference, Prentice Hall, Upper Saddle River, Nj, 1998

[2]. B. Sklar, Digital Communications: Fundamentals And Applications, Ch. 4, Englewood Cliffs, Nj: Prentice Hall, 1988.

[3]. Extensive Simulation Performance Analysis For Dsdv, Dsr And Aodv Manet Routing Protocols. Qutaibarazouqi, Ahmed Boushehri, Mohamed Gaballah, Linaalsaleh. S.L. : Ieee, 2013. 978-0-7695-4952-1.

[4]. Cross Layer Approach For Routing Protocol And. Priyankamakwana, Dhavalsinhgohil. 2, Mehsana :S.N., February 2014, International Journal Of Enhanced Research In Management & Computer Applications, Vol. 3. 2319-7471.

[5]. Effect Of Variation In Constraints On The Performance Of Manet Routing Protocol. Prem Chand, Dr. M.K. Soni. Faridabad : Ieee, 2014. 978-1-4 799-2995-5.

[6]. Performance Evaluation Of Ad-Hoc On Demand Multipath Distance Vector Routing Protocol .P.Jammulaiah, P.Ravindranaik, Ravi Gorripati.2013, Ijwcnt, Vol. 2.

[7]. A Highly Adaptive Fault Tolerant Routing Protocol For Energy Constrained Mobile Ad Hoc Networks. P. Manickam, Dr. D. Manimegalai. S.L. : Jatit, 2013, Vol. 57. Issn: 1992-8645.

[8]. James E. Gilley: "Bit-Error-Rate Simulation Using Mat Lab", Transcrypt International, Inc., August 19, 2003.

[9]. Wikipedia, Free Encyclopaedia, Article On Bit Error Rate Http://En.Wikipedia.Org/Wiki/Bit_Error_Rate.

[10]. Wikipedia, Free Encyclopaedia, Article On Signal To Noise Ratio Http://En.Wikipedia.Org/Wiki/S/N_Ratio

[11]. John. G. Proakis, "Digital Communications", Mcgraw-Hill Series In Electrical And Computer Engineering, Third Ed.

[12]. The Math Works, Inc., The Student Edition Of Matlab Version 7 User's Guide, Prentice Hall, Isbn 0-13-184979-4, 1995.

[13]. Survey Of Routing Protocols In Manet. Udaysingh A. Bagade, Suresh S. Asole. Maharashtra :S.N., April 1, 2014, Ijpret, Vol. 2. 2319-507x.

[14]. M. Jeruchim, "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.

[15]. A Survey Of Routing Protocols In Mobile Ad Hoc Networks. Sunil Taneja, Ashwani Kush. August 2010, International Journal Of Innovation, Management And Technology, Vol. 1. Issn: 2010-0248

Ahlal H Montaser Mohamed PhD in the Department of Computing atUniversity ofBradford,UK(**2004 – 2008** **)**.BSc at Department of control engineering, Faculty of Electronic Engineering, Baniwalid, Libya(**1991-1999** **)**.MSc Computing at at University of Bradford, UK.(2002-2003). **Staff Member inFaculaty ITUniversity of Tripoli(2016).**

Amamer Khalil Masoud Ahmidat PhD in the computer and information Faculaty of electrical Engand information technical University of Kosice Slovaki (**2003-2008**) . BSc electronic .EngComputer Dept (**1987-1991).**MSc in computer engineering and information Faculaty of electrical Eng.and information technical University of Kosice Slovaki (**1995-1998**).University **Staff Member in Faculaty of Science in Baniwalid**, Assistant Professor from**(01-10-2013**.)