PERFORMANCE ANALYSIS OF WIRELESS COMMUNICATION LINK USING ORTHOGONAL FDM (OFDM) OVER FADED CHANNEL AND SPACE TIME BLOCK CODE

A Thesis

Submitted to the Department of Electrical and Electronics Department

Of

BRAC University

By

Tasfeen Auyan

Student ID: 07110093

In Partial Fulfillment of the

Requirements for the Degree

Of

Bachelor of Science in Electronics and Communication Engineering

August 2010

DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of
Supervisor

Signature of
Author

ACKNOWLEDGMENTS

Thanks to the Almighty that I have finished my work properly. Special thanks to Dr. Satya Prasad Majumder who supervised me in my thesis work and taught me how to think of random data clearly for analyzing them. I am also grateful to Nazmus Saquib, Fariah Mahzabeen, Mehdi Zahid Sadi, Apurba Saha, Rumana Rahman, Zawad Hossain and Mahbub Morshed for giving their precise time which helps my work done properly.

ABSTRACT

Analysis will be carried out for an OFDM wireless communication system using space time block code (STBC) at the transmitter and considering the effect and the wireless channel like delay spread and fading. The analysis will include the effect of imperfect timing recovery at the output of the receiver. The expression for Bit Error Rate with STBC and timing error will be developed. Performance results will be evaluated numerically. Performance degradation due to imperfect timing synchronization will be determined.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1 Overview:

The target of next generation wireless communication is to achieve high data rates with low bandwidth. It should be power efficient. At present time Orthogonal Frequency Division Multiplexing is widely used for its bandwidth efficiency property. Because of its orthogonal characteristic more data can be transmitted at a certain amount of bandwidth compare to the other systems. Increasing the diversity gain is another way to achieve good performance. By using Space Time Block Code the antenna diversity gain can be increased. In this paper analysis of the non-STBC OFDM system has shown in the first part. Second part shows the analysis of STBC-OFDM. At the end of the paper equations for Signal to Noise Ratio (SNR) and Bit Error Rate (BER) have been derived analytically using four transmitting antennas and one receiving antenna and 6 transmitting antennas and one receiving antenna.

CHAPTER II

NON-STBC-OFDM

2.1 Orthogonal Frequency Division Multiplexing:

Orthogonal frequency division multiplexing (OFDM) is being considered the most promising multiplexing techniques to support the future wireless multimedia communication system. It is because of its bandwidth efficiency performance. The affects Inter Symbol interference (ISI) is also very less than the system compare to the other multiplexing techniques. OFDM has been adopted and implemented in wire and wireless communication system.

Unfortunately OFDM is very sensitive to the synchronization errors such as Carrier frequency offset (CFO), timing jitter and phase noise. The CFO arise mainly due to the Doppler shift. The effect is caused by the CFO reduce the signal amplitude and makes interference between the carriers. Phase noise resulted from the imperfection of the local oscillator (LO). Timing error would occur either when the clock signal is not correctly recovered or when sampling circuit is not perfect. Propagation delay of the IC also causes the timing error. Another term is AWGN. It is introduced in the channel through which data is transmitted.

The purpose of this paper is analysis the Signal To Noise Ratio (SNR) by changing the CFO, timing jitter, phase noise. Analysis has been done from the graphs that have been plotted with the help of MATLAB Program. Another graph also shows the Bit Error Rate according to different SNR.
The rest of the paper is organized as following sections: In section 2 CFO, timing jitter and phase noise have been described with mathematical expression. Basic OFDM system diagram is also described in this section. In section 3 the equation of the SNR has been shown in terms of CFO, phase noise, timing jitter and other parameters like input SNR (γ_{in}), number of sub carriers (N) and attenuation / gain (α) parameters.

2.2 System Model and Description:

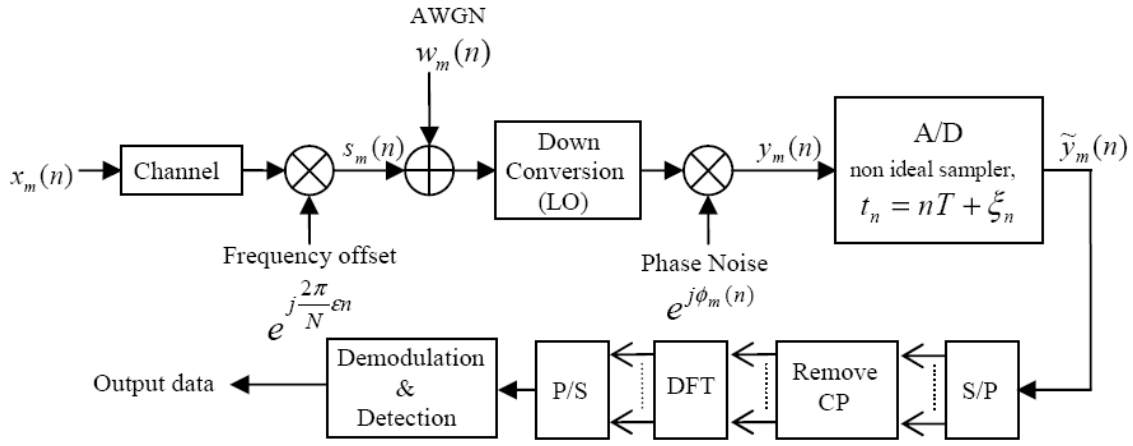

Figure 1. OFDM system model (receiver section) in the presence of CFO, phase noise and timing

Consider the m^{th} symbol of an N- sub carrier OFDM system in presence of normalized CFO (ε), phase noise $\varphi_m(n)$ and timing jitter (ξ), as shown in figure 1.

2.2.1 CFO:

The absolute value of CFO is f_ε, is either an integer multiple or a fraction of Δf. Now if the f_ε is normalized to the sub carrier spacing Δf then normalized CFO of the channel is expressed as

$$\varepsilon = \frac{f_\varepsilon}{\Delta f} = \delta + \in$$

Where δ is an integer and |ϵ|=<0.5
If the CFO occurs then the symbol transmitted on a certain sub carrier k, will shift to another sub carrier $k_s = k+\delta$.

2.2.2 Phase Noise:

Phase noise $\varphi_m(n)$ can be generated at both transmitter and receiver side. It can be modeled as

$$\varphi_m(n) = \varphi_{m-1}(N-1) + \sum_{i=-N_g}^{n} u[m(N+N_g)+i]$$

$$= \sum_{i=0}^{m(N+N_g)+N_g+n} u(i) = C_m + \sum_{i=0}^{n} u(T_m + i)$$

Where C_m and T_m are defined by $\sum_{i=0}^{m(N+N_g)+N_g-1} u(i)$ and $m(N+N_g)+N_g$ respectively. Ng is the length of cyclic prefix and u(i) denotes gaussian random variables having zero mean and variance of σ_u^2.

There is another noise that is introduced by the channel. It is Additive White Gaussian Noise (AWGN). It is added to the message signal and its PDF follows gaussian's distribution function.

2.2.3 Timing Jitter:

In the sampling circuit at the receiver additional error may occur in the determination of the best sampling phase. This means that the sampling instants are non-ideal and is given by-

$t_n = nT + \xi_n$; Where ξ_n is the timing jitter of the n_{th} sampling instant normalized by the symbol period T.

2.3 Calculation of SNR and BER:

As shown in figure: 1 the transmitted OFDM signal for the m_{th} symbol is given by the N point complex modulation sequence

$$x_m(n) = \sum_{k=0}^{N-1} X_m(k) e^{j\frac{2\pi}{N}nk}$$

Where n ranges from 0 to N+Ng-1
After passing through a rayleigh fading channel and LO, the received signal impaired by AWGN and PN can be modeled as-

$$y_m(n) = [\sum_{k=0}^{N-1} X_m(k) H_m(k) e^{j\frac{2\pi}{N}n(k+\varepsilon)}] e^{j\varphi_m(n)} + w_m(n)$$

$$\text{or,} \qquad y_m(n) = s_m(n) e^{j\varphi_m(n)} + w_m(n)$$

Where

$$s_m(n) = \sum_{k=0}^{N-1} X_m(k) H_m(k) e^{j\frac{2\pi}{N}n(k+\varepsilon)}$$

Here $H_m(k)$ is the transfer function of the rayleigh fading channel at the frequency of the k_{th} carrier and $w_m(n)$ is the complex envelope of the AWGN with zero mean and variance $\sigma^{2.}$
Assuming $\varphi_m(n)$ is very small so,

$$e^{j\varphi_m(n)} \approx 1 + j\varphi_m(n)$$

We can then write,

$$y_m(n) = s_m(n) + s_m(n) j\varphi_m(n) + w_m(n)$$

After DFT we get,

$$Y(k) = S(k) + S(k) \otimes j\Theta(k) + W(k)$$

Where S(k), Θ(k) and W(k) are the DFT responses of $s_m(n)$, $\varphi_m(n)$ and $w_m(n)$ respectively.

2.3.1 Equations For SINR and BER:

Finally SINR equation can be expressed by

$$SINR(\varepsilon, \sigma_u^2) \geq \frac{\gamma\{sinc^2(\pi\varepsilon)\}}{1+\gamma[0.5947(\sin\pi\varepsilon)^2 + \{\frac{\sigma_u^2}{2N} sinc^2(\pi\varepsilon) \sum_{r=1}^{N-1} \frac{1}{\sin^2(\frac{\pi r}{N})}\}]} \; ; \; |\varepsilon| \leq 0.5$$

This Signal To Interference plus Noise Ratio or SINR is in terms of CFO and the variance of σ^2. This is with out timing jitter (ξ). The equation for SINR is including the timing jitter (ξ) is shown below-

$$SINR(\varepsilon, \sigma_u^2, \xi) \geq \frac{\gamma(1-\xi)\{sinc^2(\pi\varepsilon)\}}{1+\gamma(1-\xi)[0.5947(\sin\pi\varepsilon)^2 + \{\frac{\sigma_u^2}{2N} sinc^2(\pi\varepsilon) \sum_{r=1}^{N-1} \frac{1}{\sin^2(\frac{\pi r}{N})}\}] + \gamma\xi} ; \quad |\varepsilon| \leq 0.5 \quad |\xi| \leq 1$$

Now the Bit Error Rate can be determined with the help of E_0/N_0 the equation below:

BER= 0.5*erfc(√(SINR))

2.4 Results (OFDM):

Table 1: System and Channel Parameters (OFDM)

No. of Sub carriers(N)	64
Channel type	Rayleigh fading
Input SNR	20dB
Channel attenuation/dB(c	Ideal (1)

2.4.1 Discussion of Results (OFDM):

In figure: 2 x-axis denotes the normalized CFO and y-axis denotes the SNR (dB). It is observed that when CFO is zero then SNR is high. And after that SNR is exponentially decreasing with the increasing of CFO. There are three plotting shows in the figure for different variance of phase noise.

In the figure: 3 x-axis denotes the variance of Phase Noise and y-axis denotes the SNR (dB). It is shown that increasing the phase noise will decrease the SNR. There are three plotting for three different value of CFO (e). The curve is decreasing if the value of CFO is increasing.

Figure: 4 shows the SNR Vs Timinig Jitter curve. It is observed that if the delay or the timing jitter increases then the signal strength decreased. For this the SNR is also decreasing. There are plotting shows different graphs for different values of phase noise and CFO.

Figure: 5 shows the SNR Vs BER graph. And Figure: 6 shows the same graph or different value of noise.

From the graph it has been seen that for v=0.5, we get highest SNR curve. And for this highest SNR value the BER is decreasing fast compared to the other value of v and corresponding SNR.

CHAPTER III

STBC-OFDM

3.1 Space Time Block Code – Orthogonal Frequency Division Multiplexing:

Severe attenuation in a multipath wireless environment makes it extremely difficult for the receiver to determine the transmitted signal unless the receiver is provided with some form of diversity i.e. some less-attenuated replica of the transmitted signal is provided to the receiver. In some applications, the only practical means of achieving diversity is deployment of antenna array at the transmitter and/or receiver end. As the current trend of communication systems demands highly power-efficient and bandwidth-efficient schemes, techniques that provide such desirable properties are considered very valuable in next generation wireless systems. Making use of multiple antennas increases the capacity of the system with the associated higher data rates than single antenna systems. Space-Time coding is a power-efficient and bandwidth-efficient method of communication over a fading channels by using multiple transmit antennas systems.

Previously in this paper the performances have been shown in non-STBC OFDM system. Author derived the equation of Signal to Noise Ratio (SNR) and Bit Error Rate (BER) for 2:1 transmission system. This paper extended the work by deriving the equations for 4:1 and 6:1 transmission system. After that performances have been shown by plotting various graphs. Results also show the effect of the Inter Carrier Interference (ICI), Channel Estimator Error, and the Additive White Gaussian Noise (AWGN). Gray code mapping is used to calculate the BER. Equations for the SNR and BER have been derived analytically.

3.2 System Model for STBC-OFDM:

We consider an OFDM system with transmit diversity, in which the total system bandwidth is divided into *N* equally spaced and orthogonal sub-carriers. We investigate the system with four transmission antennas and one receiving antenna. During the first time instant, the four symbols [X0 X1 X2 X3] are transmitted from four antennas simultaneously, with X0, X1, X2 and X3 transmitted from all four antennas. In the second time slot [-X1* X0* -X3* X2], third time slot [-X2* -X3* X0* X*] and fourth time slot [X3 -X2 –X1 X0] are transmitted.

This encoding of the transmitted symbol sequence from the transmit antennas is given by then encoding matrix

$$\begin{pmatrix} H_0 & H_1 & H_2 & H_3 \\ -H_1^* & H_0^* & -H_3^* & H_2^* \\ -H_2^* & -H_3^* & H_0^* & H_2^* \\ H_3 & -H_2 & -H_1 & H_0 \end{pmatrix}$$

For each transmit antenna, a block of *N* complex-valued data symbols ***{X(k)} for k=0 to N-1*** are grouped and converted into a parallel set to form the input to the OFDM modulator, where *k* is the sub carrier index and *N* is the number of sub carriers. The modulator consists of an Inverse Fast Fourier transform (IFFT) block. The output of the IFFT at each transmitter is the complex baseband modulated OFDM symbol in discrete time domain and is given by

$$x(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(k) e^{\frac{j2\pi kn}{N}}; \quad 0 \le n \le N-1$$

3.2.1 Channel:

The channel is modeled by a tapped delay line with channel coefficients that are assumed to be slowly varying such that they are almost constant over the two transmission instants. The channel frequency response for the *kth* subcarrier is

$$H(k) = \sum_{p=0}^{L-1} h(p) e^{\frac{j2\pi pk}{N}}$$

Where *h(p)* is the complex channel gain of the *pth* multipath component.

3.2.2 Phase Noise:

The phase noise θ(n) is modeled as a zero-mean continuous Brownian motion process with variance σ_θ^2. The phase noise increments take the form of a Wiener process, with independent Gaussian increments.

3.2.3 Received Signal:

The time-domain received signals at the first and second transmission instances at the input to the FFT block are respectively given by

$$y^0(n) = (h_0(n) \lozenge x_0(n) + h_1(n) \lozenge x_1(n) + h_2(n) \lozenge x_2(n) + h_3(n) \lozenge x_3(n) + w(n)^0) e^{j\theta(n)}$$

$$y^1(n) = (-h_0(n) \lozenge x_1^*(n) + h_1(n) \lozenge x_0^*(n) - h_2(n) \lozenge x_3^*(n) + h_3(n) \lozenge x_{2^*}(n) + w(n)^1) e^{j\theta(n)}$$

$$y^2(n) = (-h_0(n) \lozenge x_2^*(n) - h_1(n) \lozenge x_3^*(n) + h_2(n) \lozenge x_0^*(n) + h_3(n) \lozenge x_2^*(n) + w(n)^2) e^{j\theta(n)}$$

$$y^3(n) = (h_0(n) \lozenge x_3(n) - h_1(n) \lozenge x_2(n) - h_2(n) \lozenge x_1(n) + h_3(n) \lozenge x_0(n) + w(n)^3) e^{j\theta(n)}$$

Where ◊ represents linear convolution, subscripts indicate antenna index, and superscripts indicate transmission instant. The complex Gaussian random variable *w(n)* represents the Additive White Gaussian Noise (AWGN) term with $\sigma_w^2 = E[|w(n))|^2]$, and θ(n) is the phase noise.

3.2.4 Detection with Imperfect Channel Estimation:

In the presence of imperfect channel estimation, we assume a channel estimation model such that the channel estimate *H'* of the true channel *H* is given by

$$\begin{pmatrix} H_0 & H_1 & H_2 & H_3 \\ -H_1^* & H_0^* & -H_3^* & H_2^* \\ -H_2^* & -H_3^* & H_0^* & H_2^* \\ H_3 & -H_2 & -H_1 & H_0 \end{pmatrix} = \begin{pmatrix} H_0 + \varepsilon_0 & H_1 + \varepsilon_1 & H_2 + \varepsilon_2 & H_3 + \varepsilon_3 \\ -H_1^* + \varepsilon_1^* & H_0^* + \varepsilon_0^* & -H_3^* - \varepsilon_3^* & H_2^* + \varepsilon_2^* \\ -H_2^* - \varepsilon_2 & -H_3^* - \varepsilon_0 & H_0^* + \varepsilon_0^* & H_2^* + \varepsilon_0^* \\ H_3 + \varepsilon_3 & -H_2 - \varepsilon_2 & -H_1 - \varepsilon_1 & H_0 + \varepsilon_0 \end{pmatrix}$$

Where ε_0 , ε_1 , ε_2 and ε_3 are the errors in the channel estimate from the first ,second, third and fourth transmit antennas respectively, and are modeled as independent zero-mean complex Gaussian random variables with variances $2\sigma\varepsilon_0^2$, $2\sigma\varepsilon_1^2$, $2\sigma\varepsilon_2^2$ and $2\sigma\varepsilon_3^2$ respectively.

3.3 Variance:

As the noise signal has both positive and negative amplitude, it is squared, then the mean has been taken, which is variance. We consider the variance of noise for calculation.

3.3.1 Variance of Noise:

The variance of the noise *W,* after some mathematical manipulations, is given by

$$\sigma_w^{2'} = E[|W|^2] = \sigma_w^2 * \begin{pmatrix} \sum_{i=0}^{3}(\sigma_{Hi}^2 + \sigma\varepsilon_i^2) & 0 \\ 0 & \sum_{i=0}^{3}(\sigma_{Hi}^2 + \sigma\varepsilon_i^2) \end{pmatrix}$$

3.3.2 Variance of Channel Estimator Error:

The variance of Ψ is given by

$$\sigma_\psi^{2'} = E[|\Psi|^2] = \sigma_{H0}^2 * E_g * \begin{pmatrix} \sum_{i=0}^{3}(\sigma\varepsilon_i^2) & 0 \\ 0 & \sum_{i=0}^{3}(\sigma_{\varepsilon i}^2) \end{pmatrix} + \sigma_{H1}^2 * E_g * \begin{pmatrix} \sum_{i=0}^{3}(\sigma\varepsilon_i^2) & 0 \\ 0 & \sum_{i=0}^{3}(\sigma\varepsilon_i^2) \end{pmatrix} + \sigma_{H2}^2 * E_g * \begin{pmatrix} \sum_{i=0}^{3}(\sigma\varepsilon_i^2) & 0 \\ 0 & \sum_{i=0}^{3}(\sigma\varepsilon_i^2) \end{pmatrix} + \sigma_{H3}^2 * E_g * \begin{pmatrix} \sum_{i=0}^{3}(\sigma\varepsilon_i^2) & 0 \\ 0 & \sum_{i=0}^{3}(\sigma\varepsilon_i^2) \end{pmatrix}$$

3.3.3 Variance of Inter Carrier Interference:

Similarly, the variance of the ICI term β' is given by

$$\sigma_{\beta}^{2'} = E[|\beta|^2] = E_g * \begin{bmatrix} \sum_{i=0}^{3}(\sigma_{Hi}^2) & 0 \\ 0 & \sum_{i=0}^{3}(\sigma_{Hi}^2) \end{bmatrix}$$

3.4 Calculation of SNR and BER:

We present the bit error rate analysis for the case of 16QAM modulation using Gray code mapping for *($b_1b_2b_3b_4$).* It is important to note that although the presentation is only for 16QAM, the following analysis is valid for all square QAM constellations. The conditional BER for bit *b1,* condition on *H0, H1, H2, H3* is given by

$$P1(b1|H0,H1,H2,H3) = \frac{1}{2} * [Q\left(\sqrt{\frac{\left(2*\frac{Eg}{5}\right)(|H0|^2+|H1|^2+|H2|^2+|H3|^2)}{6\psi^2+6\beta^2+6w^2}}\right)$$

$$+Q\left(\sqrt{\frac{\left(2*\frac{Eg}{5}\right)(|H0|^2+|H1|^2+|H2|^2+|H3|^2)}{6\psi^2+6\beta^2+6w^2}}\right)]$$

and for bit *b3 i*s given by

$$P1(b1|H0,H1,H2,H3)$$

$$= \frac{1}{2} * \left[Q\left(\sqrt{\frac{\left(9 * (\frac{2 * Eg}{5}\right)(|H0|^2 + |H1|^2 + |H2|^2 + |H3|^2)}{6\psi^2 + 6\beta^2 + 6w^2}} \right) \right.$$

$$+ Q\left(\sqrt{\frac{\left(2 * \frac{Eg}{5}\right)(|H0|^2 + |H1|^2 + |H2|^2 + |H3|^2)}{6\psi^2 + 6\beta^2 + 6w^2}} \right)$$

$$+ Q\left(\sqrt{\frac{\left(2 * \frac{Eg}{5}\right)(|H0|^2 + |H1|^2 + |H2|^2 + |H3|^2)}{6\psi^2 + 6\beta^2 + 6w^2}} \right)$$

$$+$$

$$\left. Q\left(\sqrt{\frac{\left(25 * (\frac{2 * Eg}{5}\right)(|H0|^2 + |H1|^2 + |H2|^2 + |H3|^2)}{6\psi^2 + 6\beta^2 + 6w^2}} \right) \right]$$

From which the SNR γ is given by

$$\gamma = \frac{(|H0|^2 + |H1|^2 + |H2|^2 + |H3|^2)}{(6\psi^2 + 6\beta^2 + 6w^2)} \qquad *2*Eg/5$$

It follows a Chi-square distribution with probability density function (PDF) given by

$$P(\gamma) = \frac{1}{2*6\gamma^2} * \exp\left[-\frac{\gamma}{2*6\gamma^2}\right]$$

Due to the symmetry of square M-QAM constellations, the BER for the in-phase and quadrature bits are equal such that *Pe (b*1) = *Pe (b2)* and *Pe (b3)= Pe (b4*). Therefore the average BER is obtained by averaging the conditional BER of b1 and *b3* over the PDF of the SNR γ. The average BER is therefore given by

$$Pe = \frac{1}{2} * \int_0^{\infty} [Pe(b1|H0,H1,H2,H3) + Pe(b3|H0,H1,H2,H3)\ ]p(\gamma)d(\gamma)$$

Similarly for 6:1 transmission system SNR γ is given by

$$\gamma = \frac{(|H0|^2+|H1|^2+|H2|^2+|H3|^2+|H4|^2+|H5|^2)}{(6\psi^2+6\beta^2+6w^2)} \qquad *2*Eg/5$$

And BER is giver by

$$Pe = \frac{1}{2} * \int_0^{\infty} [Pe(b1|H0,H1,H2,H3,H4,H5) + Pe(b3|H0,H1,H2,H3,H4,H5)\ ]p(\gamma)d(\gamma)$$

3.5 Results (STBC-OFDM):

The parameters that uses in the calculation are shown in table-2

Table 2: System and Channel Parameters (STBC-OFDM)

Parameters	Values
σh^2	0.1 0.2 0.02 0.4
$\sigma \varepsilon^2$	0.1 0.04 0.06
Subcarriers (N)	64
Channel Path Gains	-9.7 -0.9 -8.5 -0.5

3.5.1 Discussion of Results (STBC-OFDM):

Figure: 7 shows the SNR Vs BER graph for 2:1 transmission system. Figure: 8 shows the same graph for different value of noises.

Then figure: 9 shows the graph of SNR Vs BER for 4:1 transmission system along with 2:1 and 1:1 system.

It is seen that 4:1 graph is closer to the two axes than the others. It means that the BER is decreasing fast when the number of antennas increases.

Figure: 10 shows the SNR Vs BER graph of 4:1 for different value of noises. Then figure: 11 shows the performance of 6:1 along with 4:1, 2:1, 1:1 transmission system.

Figure: 12 shows the SNR Vs BER graph of 6:1 for different value of noise.

From the figure: 11 receiver sensitivity graphs are plotted for different value of noises. It shows that transmission power decreases when number of antennas increase. The graph is shown in figure: 13.

Another analysis can be drawn from the graph shown in figure: 13 is that for a fixed value of transmission power the noise term can be reduced by increasing the number of antennas.

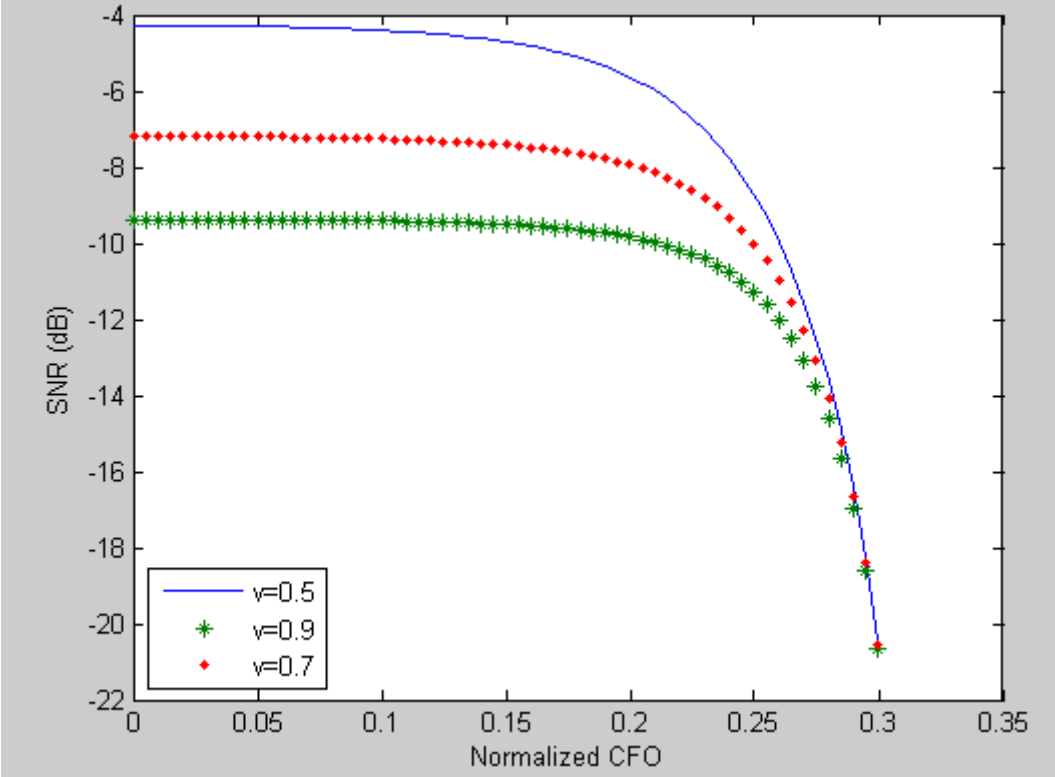

Figure 2: Graph shows SNR Vs Normalized form of CFO

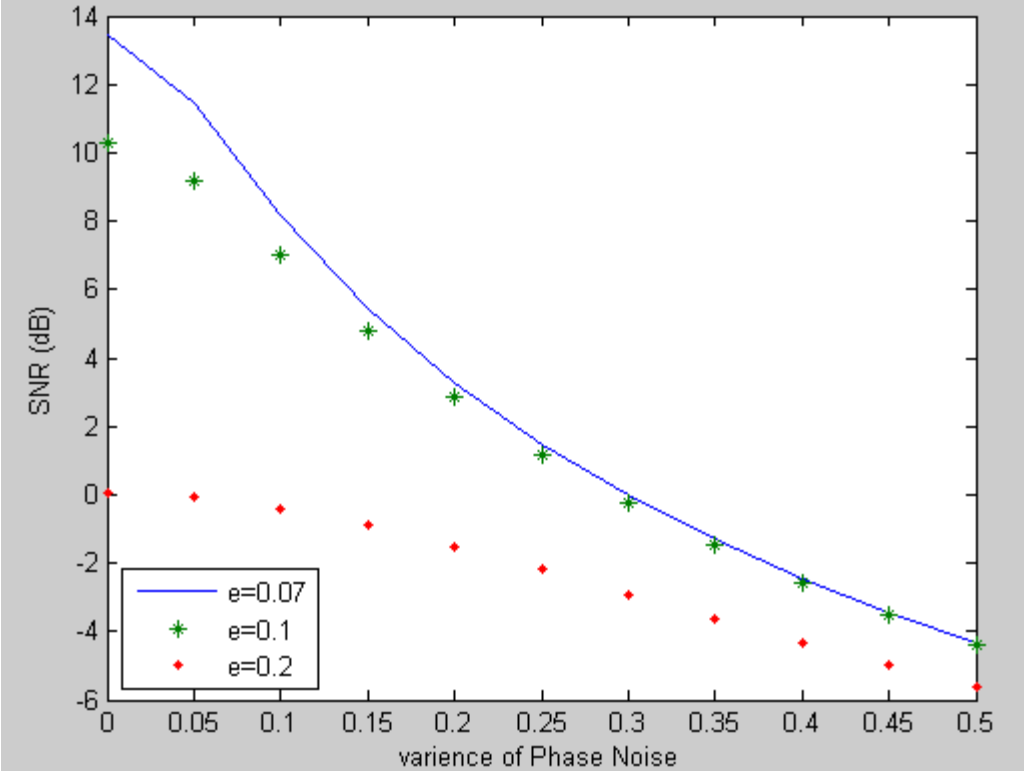

Figure 3: Graph shows SNR Vs variance of Phase Noise

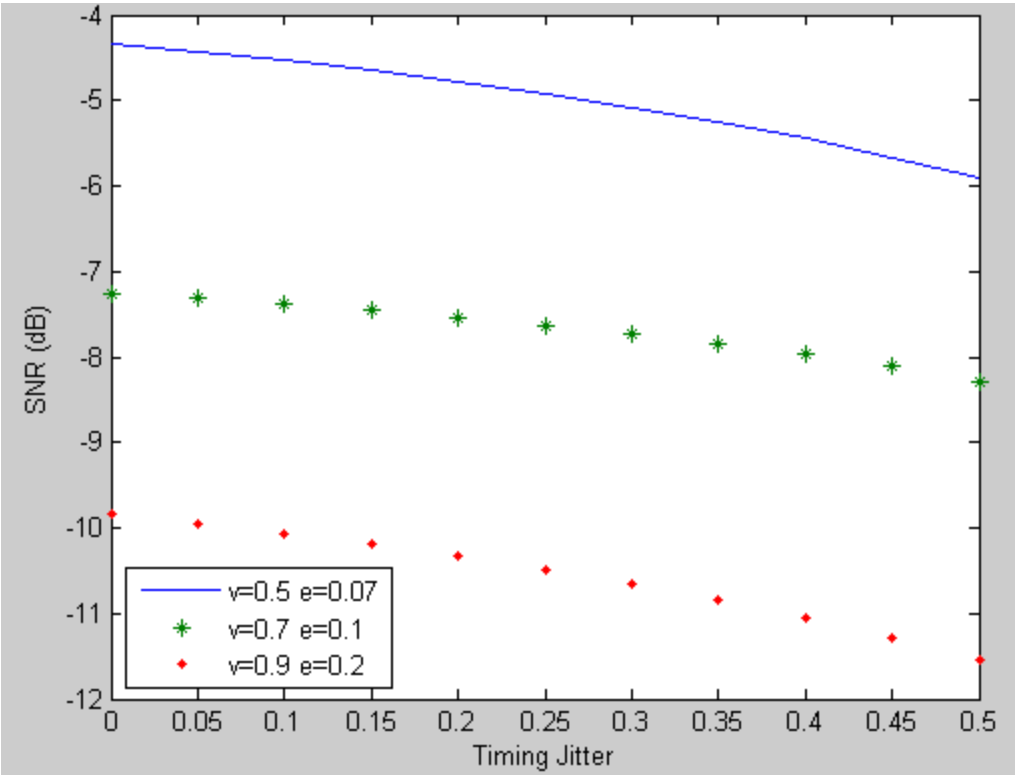

Figure 4: Graph shows SNR Vs Timing Jitter

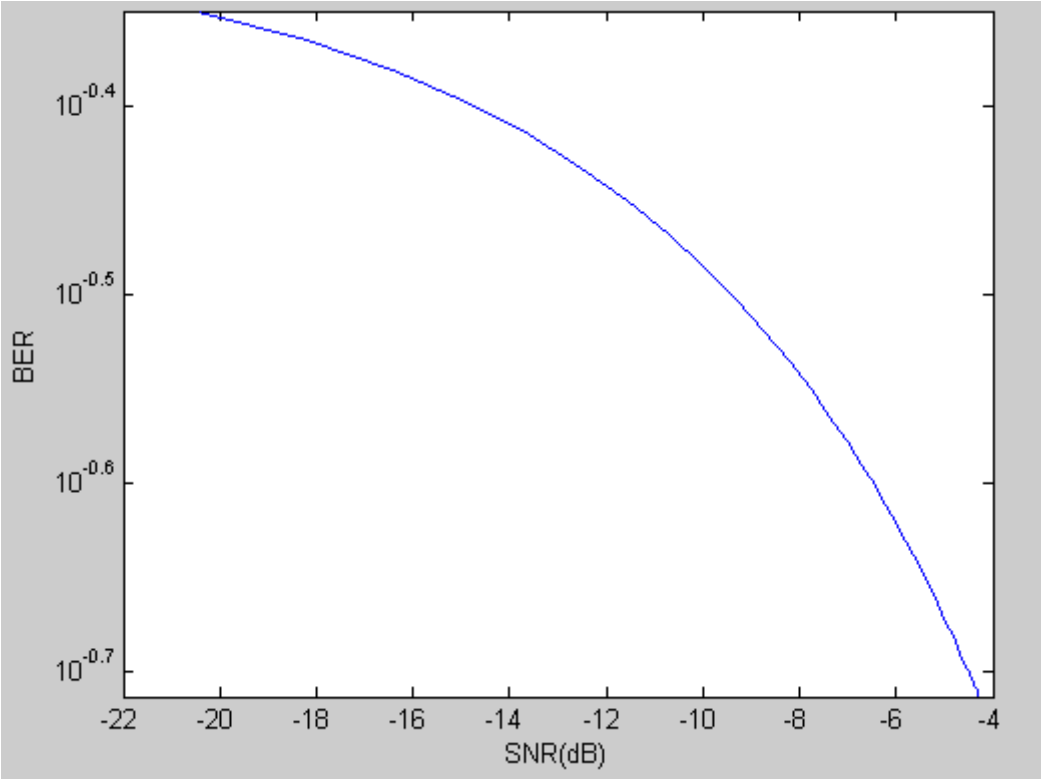

Figure 5: Graph shows the BER Vs SNR (dB)

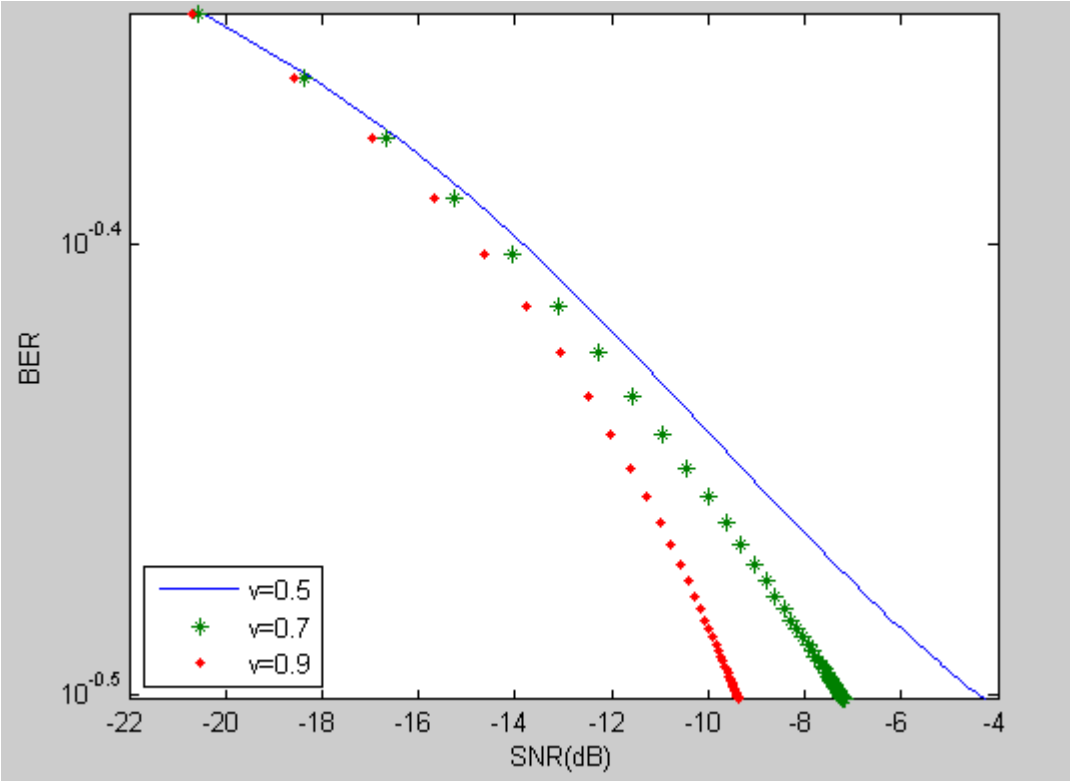

Figure 6: Graph shows different plotting of BER Vs SNR of different value of variance of Phase Noise

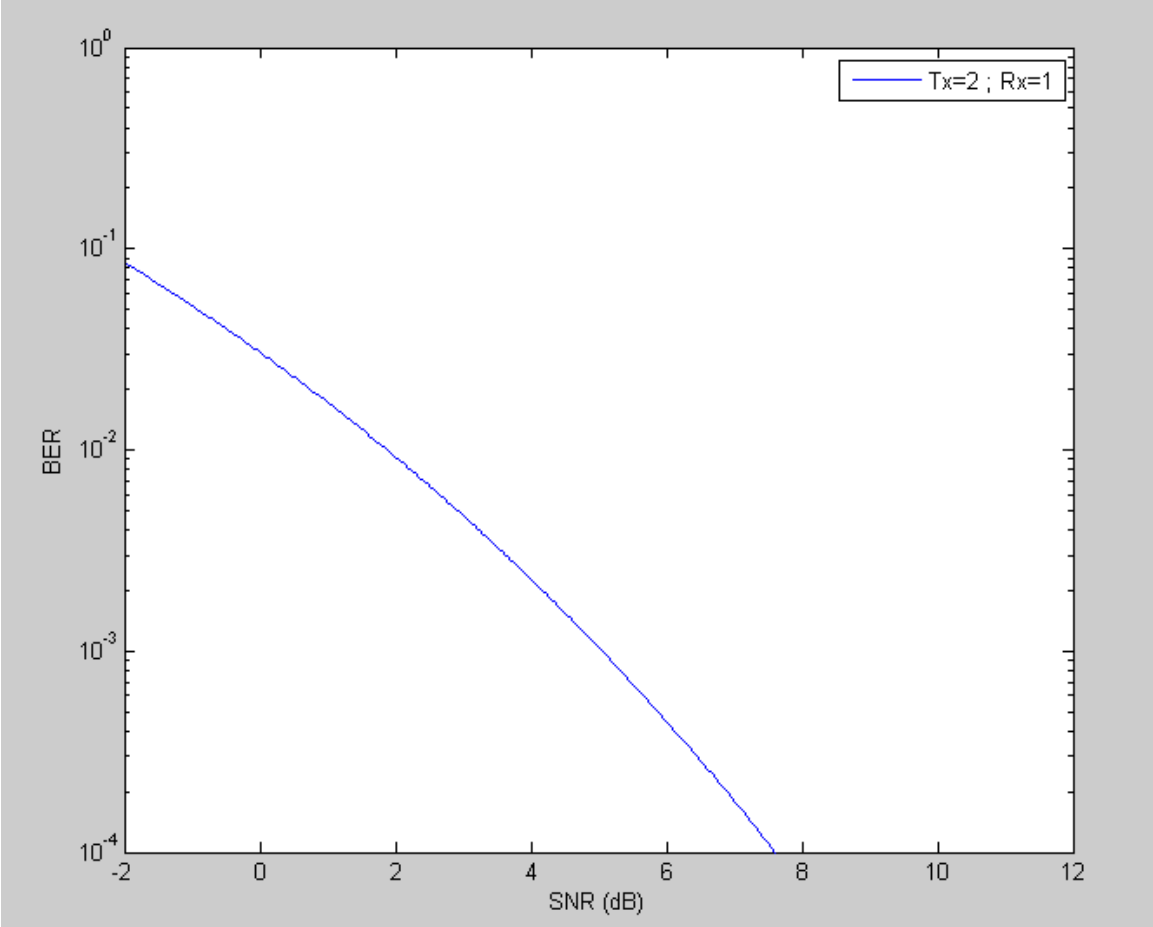

Figure 7: SNR Vs BER graph for 2:1 transmission system

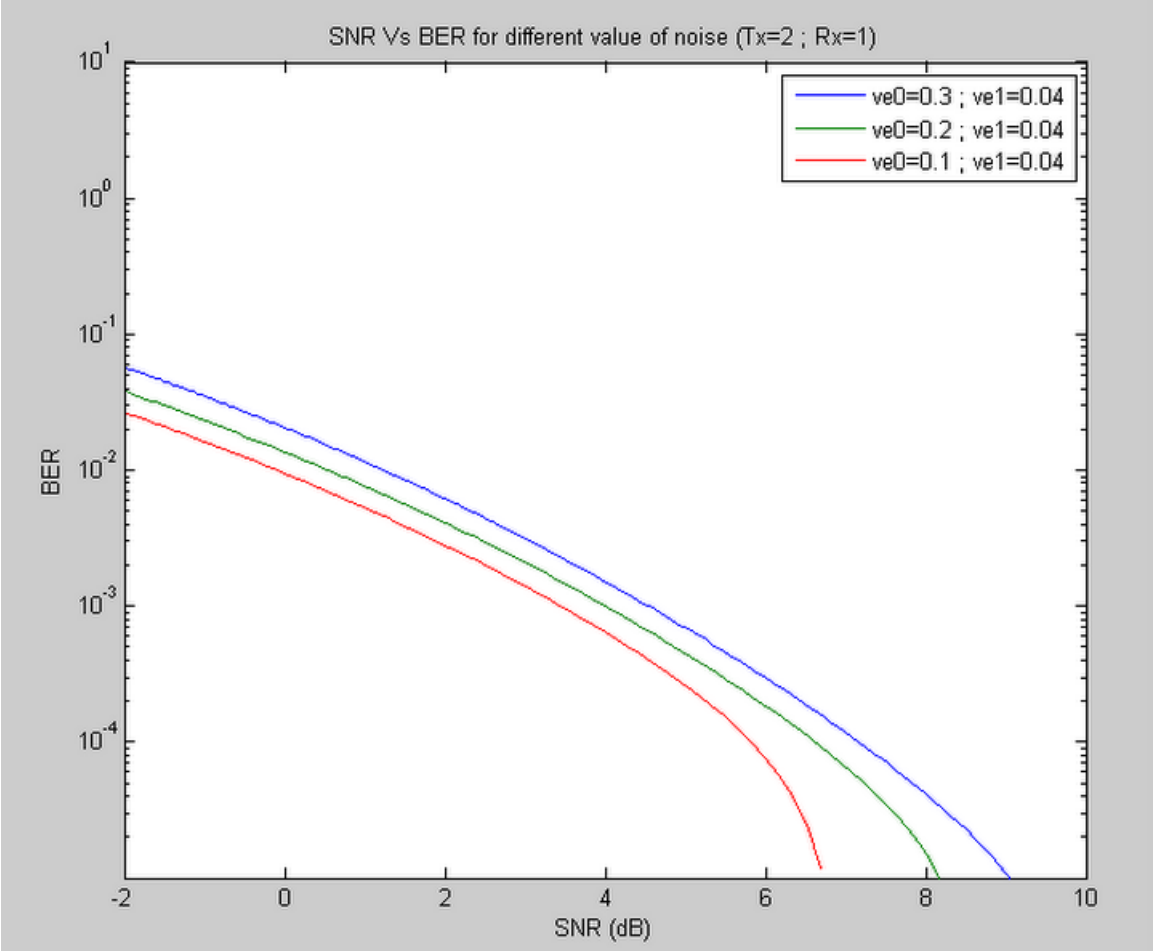

Figure 8: SNR Vs BER graph for different value of noises (Tx=2 ; Rx=1)

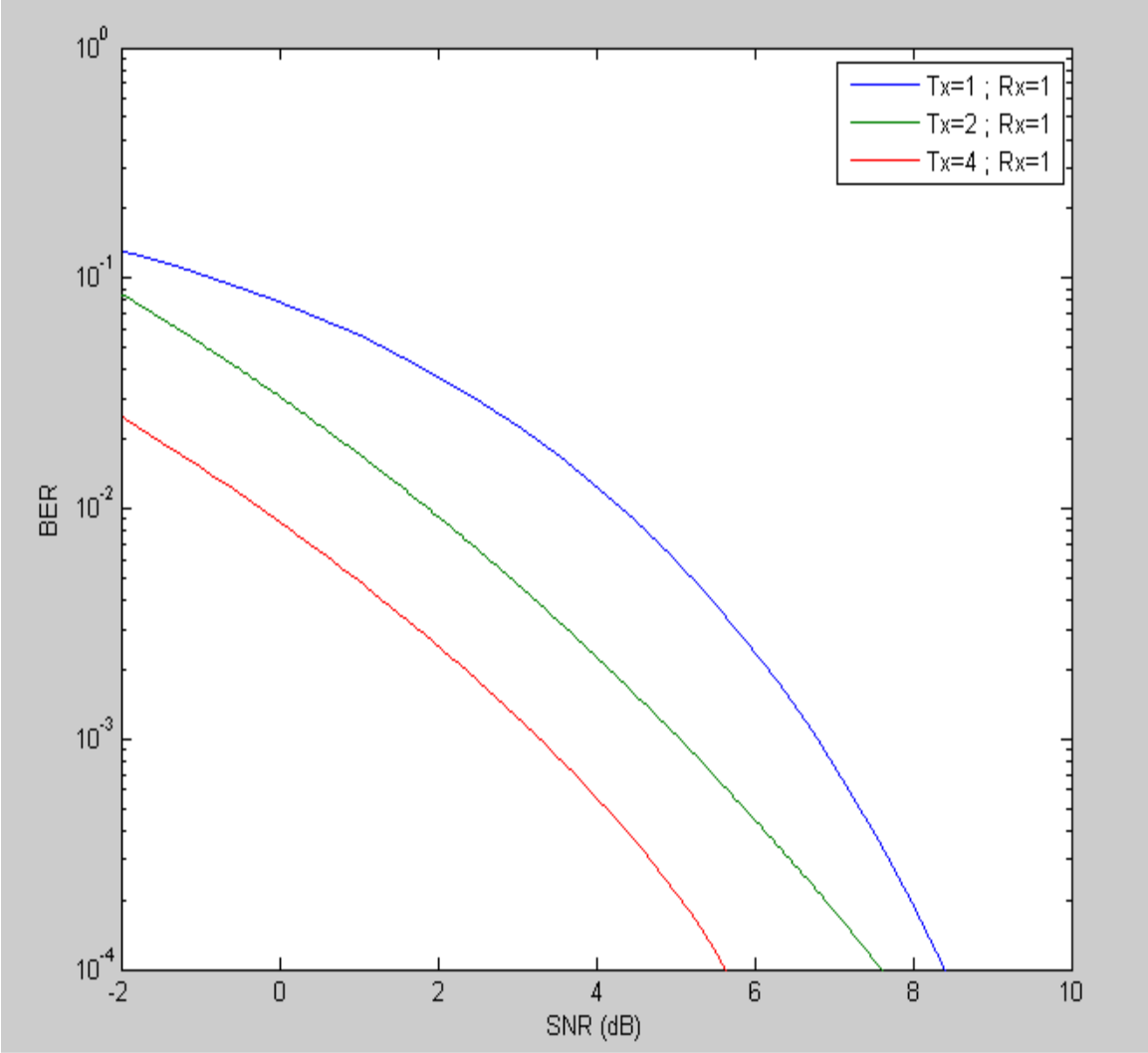

Figure 9: SNR Vs BER for 4:1, 2:1, 1:1 transmission system

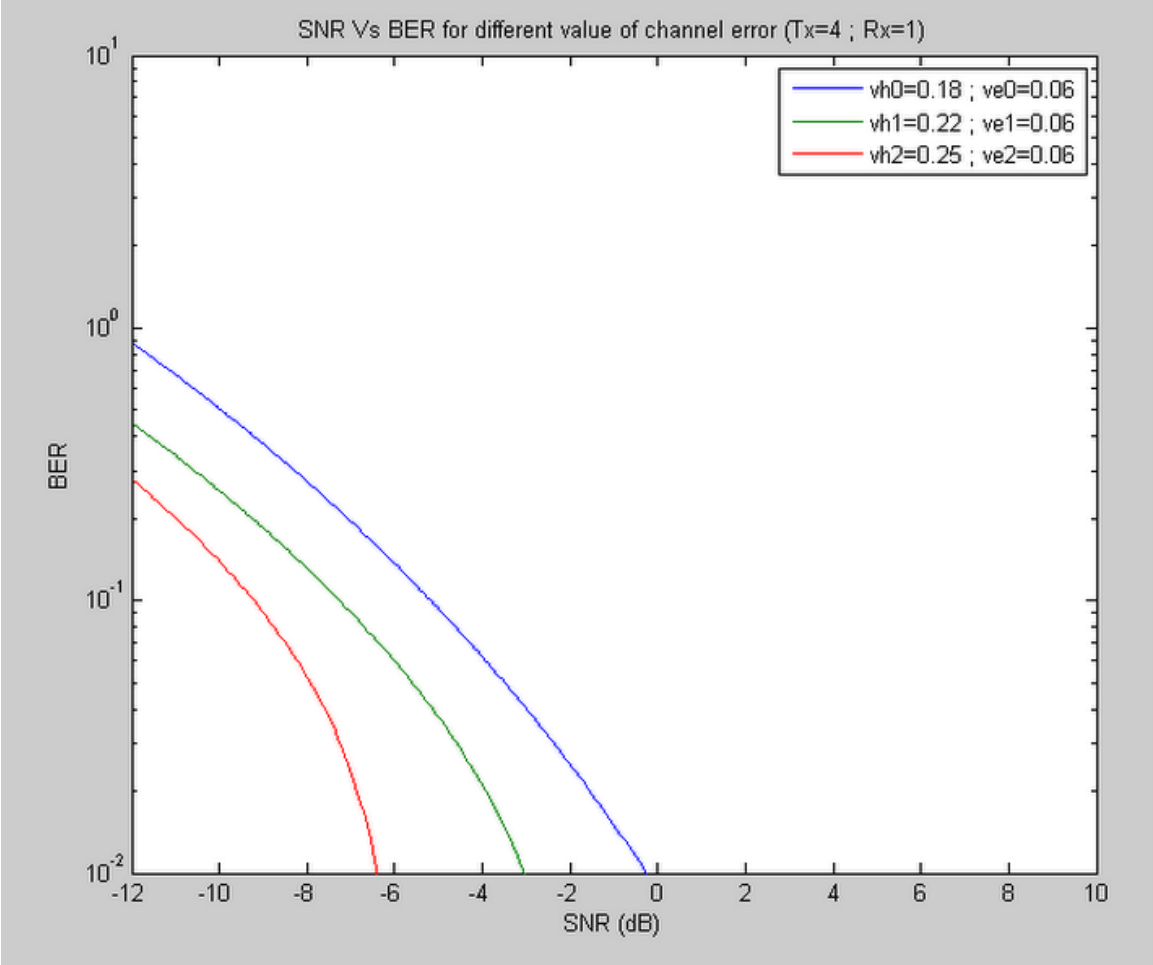

Figure 10: SNR Vs BER graph for different value of noises (Tx=4 ; Rx=1)

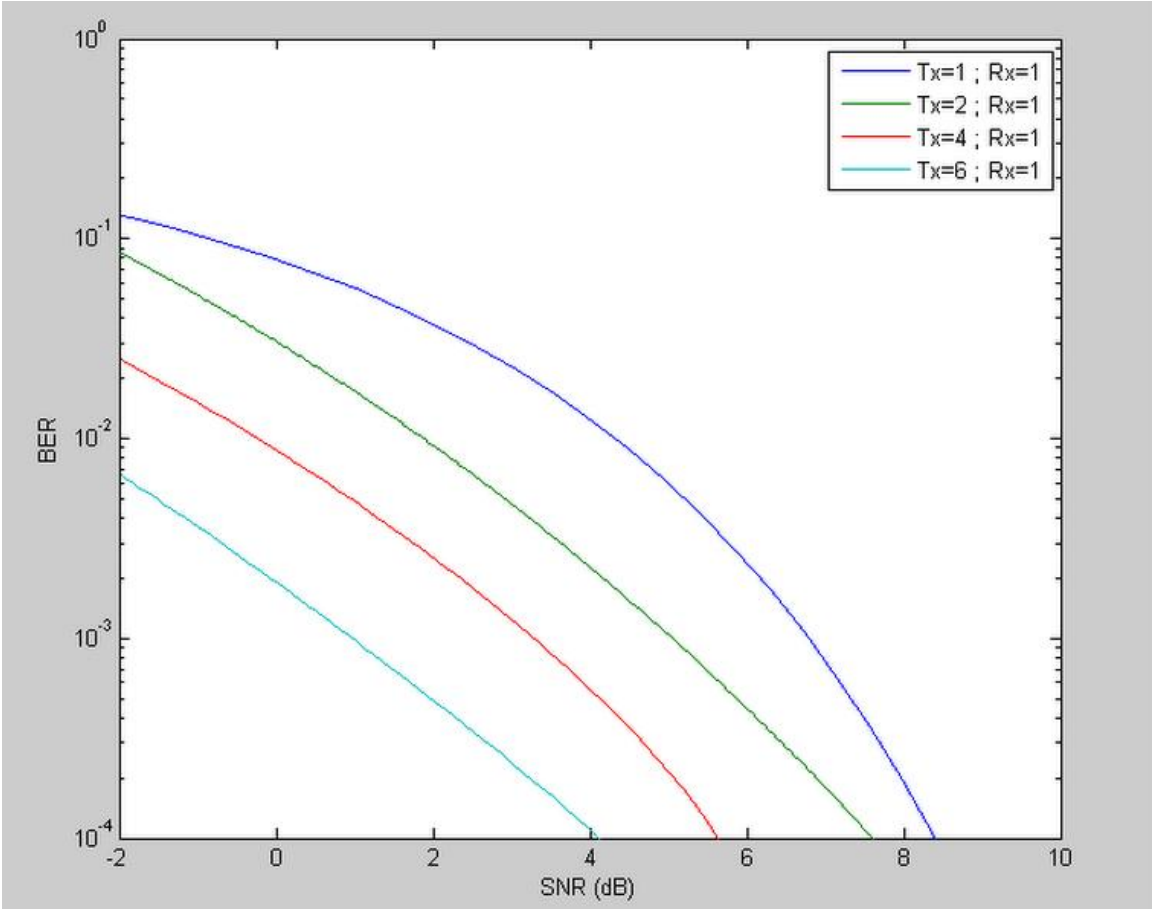

Figure 11: SNR Vs BER for 6:1, 4:1, 2:1, 1:1 transmission system

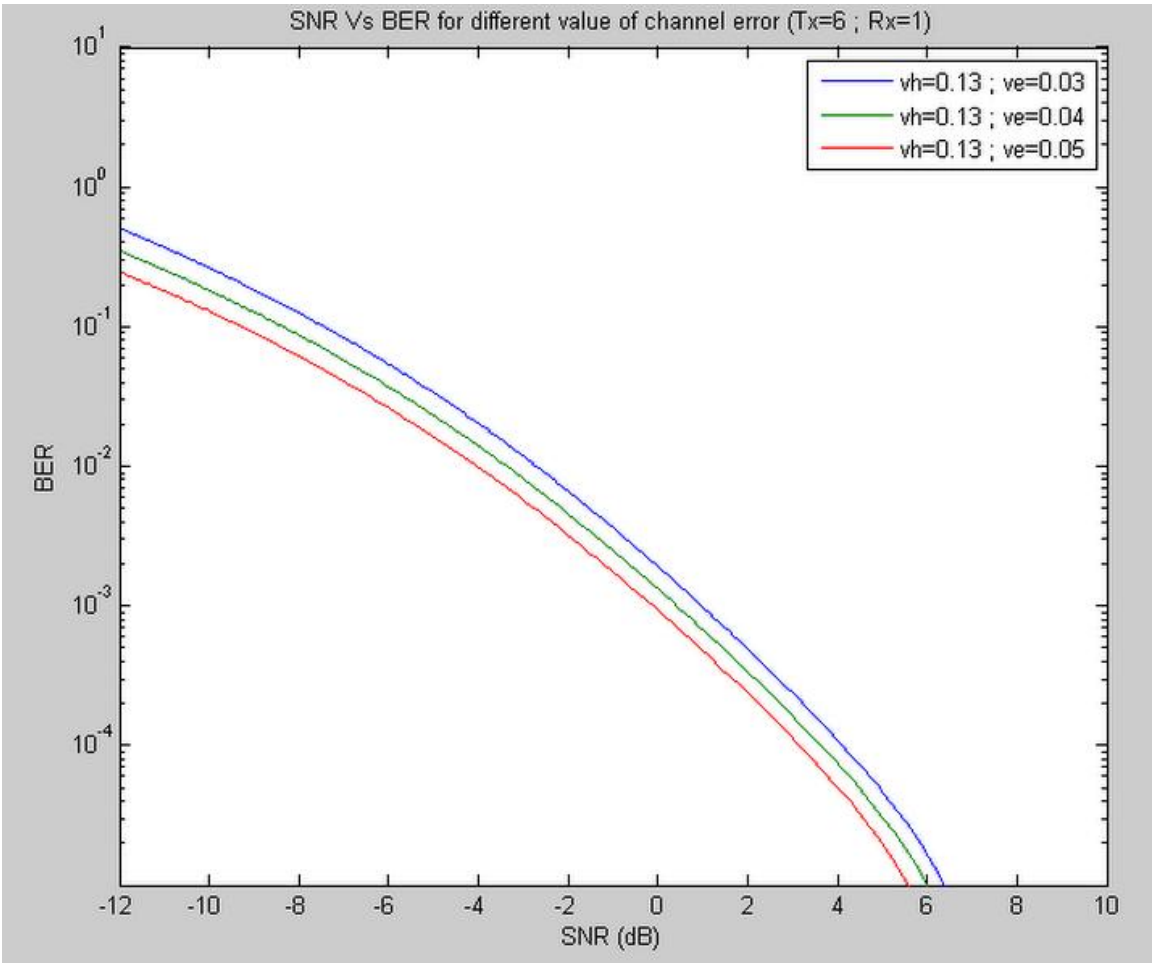

Figure 12: SNR Vs BER graph for different value of noises (Tx=6 ; Rx=1)

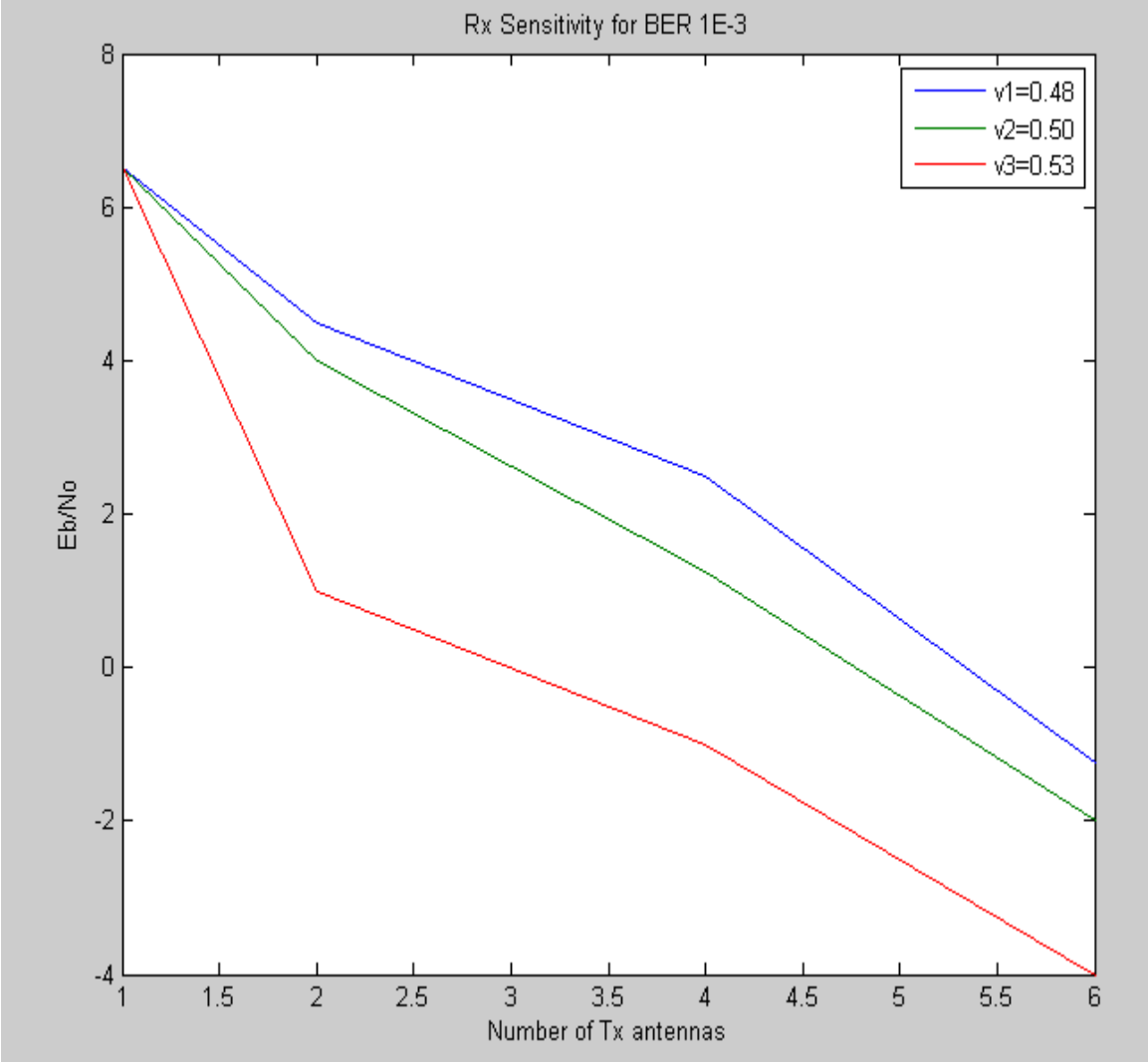

Figure 13: Rx Sensitivity for BER 1e-3

CHAPTER IV

CONCLUSION

The equations for SNR and BER using 4:1 and 6:1 transmission system are derived. The effects of noise, carrier interference and channel estimator error on the system are analyzed. The SNR Vs BER curve shows that increasing the diversity gain improve the performance of the system. Receiver sensitivity graph shows the power efficiency characteristic of the system.

LIST OF REFERENCES

[1] Shankhanaad Mallick and Dr. Satya Prasad Majumder- “Performance Analysis of an OFDM System in the Presence of Carrier Frequency Offset, Phase Noise and Timing Jitter over Rayleigh Fading Channels” ICECE,2008.

[2] Mehedi Hasan and Dr. Satya Prasad Majumder- “Performance Limitation of SIMO OFDM Wireless Link Impaired by Carrier Frequency Offset, Phase Noise and Fading”

[3] Mohamed Jalloh, Pankaj Das- “Performance Analysis of STBC-OFDM Transmit Diversity with Phase Noise and Imperfect Channel Estimation”

[4] Siavash M. Alamouti- "A Simple Transmit Diversity Technique for Wireless Communications" IEEE Journal On Selected Areas in Communications, vo1.16, pp 1454, Oct 1998.

[5] Vahid Tarokh, Hamid Jafarkhani, A. R. Calderbank- "Space-Time Block Codes from Orthogonal Designs" IEEE Transactions on Information Theory, vol.45, pp. 1456-1467, Jul 1999.

[6] Vahid Tarokh, Ayman Naguib, Nambi Seshadri, and A. R. Calderbank- "Space-Time Codes for High Data Rates Wireless Communication: Performance Criteria in the Presence of Channel Estimation Errors. Mobility, and Multipath" IEEE Transactions on Communications, vo1.47, pp. 199207, Feb 1999

[7] T.N. Zogakis and J.M. Cioffi- “The effect of timing jitter on the performance of a discrete multitone system”, IEEE Trans. Commun. Vol. 44, no. 7, pp. 799-808, July 1996.

[8] T. Pollet and M. Moeneclacy- “Synchronizability of OFDM signals”, in Proc. Globecom’95, pp. 2054-2058, Singapore, Nov. 1995.

[9] P.H. Moose- “A technique for Orthogonal Frequency Division Multiplexing Frequency Offset Correction” IEEE, Trans. Commun, vol. 42, pp. 2908-2914, Oct. 1994.
[10] S. Wu and Y. Bar-Ness- “OFDM Systems in the Presence of Phase Noise: Consequences and Solutions” IEEE Trans. Commun. Vol. 52, No. 11, pp. 1988-1996, Nov. 2004.

MATLAB CODES

%% Code 01: Code for SNR Vs Normalized form of CFO (OFDM):

```
clc;
close all;
N=64;
y=0;
for r=1:N-1
    m= 1/((sin(pi*r/N))^2);
    y=y+m;
end
v=0.5;
g=10^2;
a=1;
k=1;
for e=0:.005:.3
    d1=(v^2)*((sinc(pi*e))^2)*y/(2*N);
    d2=0.5947*((sin(pi*e))^2)
    d3=d1+d2;
    d=1+(g*(a^2)*d3);
    u=g*(a^2)*(sinc(pi*e)^2);
    s(k)=10*log10(u/d)
    k=k+1;
end
v=0.7;
k=1;
for e=0:.005:.3
    d1=(v^2)*((sinc(pi*e))^2)*y/(2*N);
    d2=0.5947*((sin(pi*e))^2)
    d3=d1+d2;
    d=1+(g*(a^2)*d3);
    u=g*(a^2)*(sinc(pi*e)^2);
    s2(k)=10*log10(u/d)
    k=k+1;
end
v=0.9;
k=1;
for e=0:.005:.3
    d1=(v^2)*((sinc(pi*e))^2)*y/(2*N);
    d2=0.5947*((sin(pi*e))^2)
    d3=d1+d2;
    d=1+(g*(a^2)*d3);
    u=g*(a^2)*(sinc(pi*e)^2);
    s1(k)=10*log10(u/d)
```

```
    k=k+1;
end
e=0:.005:.3
plot(e,s,'-',e,s1,'*',e,s2,'.');
h=legend('v=0.5','v=0.9','v=0.7',3);
set(h,'Interpreter','none');
ylabel('SNR (dB)')
xlabel('Normalized CFO')
```

%% Code 02: Code for SNR (dB) Vs Phase Noise (OFDM):

```
clc;
close all;
N= 64;
y=0;
for r=1:N-1
    m= 1/((sin(pi*r/N))^2)
    y=y+m
end
g=10^2;
a=1;
e=0.07;
k=1;
for v=0:0.05:0.5
d1=(v^2)*((sinc(pi*e))^2)*y/(2*N)
d2=0.5947*(sin(pi*e)^2)
d3=d1+d2
d=1+(g*(a^2)*d3)
u=g*(a^2)*(sinc(pi*e)^2)
s(k)=10*log10(u/d)
k=k+1;
end

e=0.1;
k=1;
for v=0:0.05:0.5
d1=(v^2)*((sinc(pi*e))^2)*y/(2*N)
d2=0.5947*(sin(pi*e)^2)
d3=d1+d2
d=1+(g*(a^2)*d3)
u=g*(a^2)*(sinc(pi*e)^2)
s1(k)=10*log10(u/d)
k=k+1;
end
```

```
e=0.2;
k=1;
for v=0:0.05:0.5
d1=(v^2)*((sinc(pi*e))^2)*y/(2*N)
d2=0.5947*(sin(pi*e)^2)
d3=d1+d2
d=1+(g*(a^2)*d3)
u=g*(a^2)*(sinc(pi*e)^2)
s2(k)=10*log10(u/d)
k=k+1;
end

v=0:0.05:0.5;
plot(v,s,'-',v,s1,'*',v,s2,'.')
h=legend('e=0.07','e=0.1','e=0.2',3);
set(h,'Interpreter','none');
ylabel('SNR (dB)');
xlabel('varience of Phase Noise');
```

%% Code 03: Code for SNR (dB) Vs Timing Jitter (OFDM):

```
clc;
close all;
N=64;
y=0;
for r=1:N-1
   m= 1/((sin(pi*r/N))^2);
   y=y+m;
end
v=0.5;
e=0.07;
g=10^2;
a=1;
k=1;
for z=0:.05:0.5
    d1=(v^2)*((sinc(pi*e))^2)*y/(2*N);
     d2=0.5947*((sin(pi*e))^2);
     d3=d1+d2;
     d4= g*(a^2)*(1-z)*d3;
     d=1+d4+(g*(a^2)*z);
     u=g*(a^2)*(1-z)*(sinc(pi*e)^2);
     s(k)=10*log10(u/d);
     k=k+1;
end
```

```
v=0.7;
e=0.1;
k=1;
for z=0:.05:0.5
    d1=(v^2)*((sinc(pi*e))^2)*y/(2*N);
     d2=0.5947*((sin(pi*e))^2);
     d3=d1+d2;
     d4= g*(a^2)*(1-z)*d3;
     d=1+d4+(g*(a^2)*z);
     u=g*(a^2)*(1-z)*(sinc(pi*e)^2);
     s1(k)=10*log10(u/d);
     k=k+1;
end

v=0.9;
e=0.2;
k=1;
for z=0:.05:0.5
    d1=(v^2)*((sinc(pi*e))^2)*y/(2*N);
     d2=0.5947*((sin(pi*e))^2);
     d3=d1+d2;
     d4= g*(a^2)*(1-z)*d3;
     d=1+d4+(g*(a^2)*z);
     u=g*(a^2)*(1-z)*(sinc(pi*e)^2);
     s2(k)=10*log10(u/d);
     k=k+1;
end
z=0:.05:0.5;
plot(z,s,'-',z,s1,'*',z,s2,'.')
h=legend('v=0.5 e=0.07','v=0.7 e=0.1','v=0.9 e=0.2',3);
set(h,'Interpreter','none');
ylabel('SNR (dB)')
xlabel('Timing Jitter')
```

%% Code 04: Code for SNR (dB) Vs BER (OFDM):

```
clc;
close all;
N=64;
y=0;
for r=1:N-1
   m= 1/((sin(pi*r/N))^2);
   y=y+m;
end
v=0.5;
```

```
g=10^2;
a=1;
k=1;
for e=0:.005:.3
   d1=(v^2)*((sinc(pi*e))^2)*y/(2*N);
   d2=0.5947*((sin(pi*e))^2)
   d3=d1+d2;
   d=1+(g*(a^2)*d3);
   u=g*(a^2)*(sinc(pi*e)^2);
   sdb(k)=10*log10(u/d);
   s(k)=(u/d);
   p(k)= 0.5*erfc(sqrt(s(k)))
   k=k+1;
end
e=0:.005:.3
semilogy(sdb,p)
ylabel('BER');
xlabel('SNR(dB)');
```

%% Code 05: Code for SNR (dB) Vs BER for different values of Phase Noise (OFDM):

```
clc;
close all;
N=64;
y=0;
for r=1:N-1
   m= 1/((sin(pi*r/N))^2);
   y=y+m;
end
v=0.5;
g=10^2;
a=1;
k=1;
for e=0:.005:.3
   d1=(v^2)*((sinc(pi*e))^2)*y/(2*N);
   d2=0.5947*((sin(pi*e))^2)
   d3=d1+d2;
   d=1+(g*(a^2)*d3);
   u=g*(a^2)*(sinc(pi*e)^2);
   sdb1(k)=10*log10(u/d);
   s(k)=(u/d);
   p(k)= 0.5*erfc(sqrt(s(k)))
   k=k+1;
end
```

```
v=0.7;
g=10^2;
a=1;
k=1;
for e=0:.005:.3
    d1=(v^2)*((sinc(pi*e))^2)*y/(2*N);
    d2=0.5947*((sin(pi*e))^2)
    d3=d1+d2;
    d=1+(g*(a^2)*d3);
    u=g*(a^2)*(sinc(pi*e)^2);
    sdb2(k)=10*log10(u/d);
    s(k)=(u/d);
    p(k)= 0.5*erfc(sqrt(s(k)))
    k=k+1;
end


v=0.9;
g=10^2;
a=1;
k=1;
for e=0:.005:.3
    d1=(v^2)*((sinc(pi*e))^2)*y/(2*N);
    d2=0.5947*((sin(pi*e))^2)
    d3=d1+d2;
    d=1+(g*(a^2)*d3);
    u=g*(a^2)*(sinc(pi*e)^2);
    sdb3(k)=10*log10(u/d);
    s(k)=(u/d);
    p(k)= 0.5*erfc(sqrt(s(k)))
    k=k+1;
end

e=0:.005:.3
semilogy(sdb1,p,'-',sdb2,p,'*',sdb3,p,'.')
ylabel('BER');
xlabel('SNR(dB)');
```

%% Code 06: Code for SNR Vs BER graph for 2:1 transmission system:

```
clc;
clear all;
close all;

vh02=0.1;
vh12=0.2;

ve02=0.02;
ve12=0.04;

eg2=5;

h02=10^(-9.7/10);
h12=10^(-0.9/10);

vsnr2=0.2;

d2=0.02;

x2=vh02+vh12+ve02+ve12;

k=1;
for vw2=0.1:0.03:10
vw12(k)=vw2*det([x2 0;0 x2])   %noise

y2=ve02+ve12;
vc2=(vh02*eg2*det([y2 0;0 y2]))+(vh12*eg2*det([y2 0;0 y2]))   %imperfect
channel estimator

z2=vh02+vh12;
vb2=eg2*det([z2 0;0 z2])   %ICI

n2=vw12+vc2+vb2
u12=(2*eg2/5)*((h02^2)+(h12^2));
q12(k)=sqrt(u12/n2(k));
qf12=qfunc(q12);
u22=9*u12;
q22(k)=sqrt(u22/n2(k));
qf22=qfunc(q22);
p12=0.5*(qf12+qf22);
q32(k)= u22/n2(k);
q42(k)= u12/n2(k);
q52(k)= 25*u12/n2(k);
```

```
qf32=qfunc(q32);
qf42=qfunc(q42);
qf52=qfunc(q52);
p22=0.5*(qf32+qf42+qf42+qf52);
p2=p12+p22;

snr2(k)=((((h02^2)+(h12^2))*2*eg2/5)/n2(k));
snrdb2(k)=10*log(snr2(k));

psnr2(k)= 0.5*(1/vsnr2)*( exp( -(snr2(k)/2/vsnr2) ) );
b12(k)=p2(k)*psnr2(k);

BER2(k)=0.5*trapz(b12)*d2;

k=k+1;
end



semilogy(snrdb2,BER2)
```

%% Code 07: Code for SNR Vs BER graph for different value of noises
%% (Tx=2 ; Rx=1):

```
clc;
clear all;
close all;
vh0=0.1;
vh1=0.2;
ve0=0.1;
ve1=0.04;
eg=5;
h0=10^(-9.7/10);
h1=10^(-0.9/10);
vsnr=0.2;
d=0.02;
x=vh0+vh1+ve0+ve1;
k=1;
for vw=0.1:0.03:10
vw1(k)=vw*det([x 0;0 x])   %noise


y=ve0+ve1;
vc=(vh0*eg*det([y 0;0 y]))+(vh1*eg*det([y 0;0 y]))   %imperfect channel estimator
```

```
z=vh0+vh1;
vb=eg*det([z 0;0 z])   %ICI

n=vw1+vc+vb
u1=(2*eg/5)*((h0^2)+(h1^2));
q1(k)=sqrt(u1/n(k));
qf1=qfunc(q1);
u2=9*u1;
q2(k)=sqrt(u2/n(k));
qf2=qfunc(q2);
p1=0.5*(qf1+qf2);
q3(k)= u2/n(k);
q4(k)= u1/n(k);
q5(k)= 25*u1/n(k);
qf3=qfunc(q3);
qf4=qfunc(q4);
qf5=qfunc(q5);
p2=0.5*(qf3+qf4+qf4+qf5);
p=p1+p2;

snr(k)=((((h0^2)+(h1^2))*2*eg/5)/n(k));
snrdb(k)=10*log(snr(k));

psnr(k)= 0.5*(1/vsnr)*( exp( -(snr(k)/2/vsnr) ) );
b1(k)=p(k)*psnr(k);

BER(k)=0.5*trapz(b1)*d;

k=k+1;
end



vh02=0.1;
vh12=0.2;
ve02=0.2;
ve12=0.04;
eg2=5;
h02=10^(-9.7/10);
h12=10^(-0.9/10);
vsnr2=0.2;
d2=0.02;
x2=vh02+vh12+ve02+ve12;
k=1;
for vw2=0.1:0.03:10
vw12(k)=vw2*det([x2 0;0 x2])   %noise
```

```
y2=ve02+ve12;
vc2=(vh02*eg2*det([y2 0;0 y2]))+(vh12*eg2*det([y2 0;0 y2]))   %imperfect
channel estimator

z2=vh02+vh12;
vb2=eg2*det([z2 0;0 z2])   %ICI

n2=vw12+vc2+vb2
u12=(2*eg2/5)*((h02^2)+(h12^2));
q12(k)=sqrt(u12/n2(k));
qf12=qfunc(q12);
u22=9*u12;
q22(k)=sqrt(u22/n2(k));
qf22=qfunc(q22);
p12=0.5*(qf12+qf22);
q32(k)= u22/n2(k);
q42(k)= u12/n2(k);
q52(k)= 25*u12/n2(k);
qf32=qfunc(q32);
qf42=qfunc(q42);
qf52=qfunc(q52);
p22=0.5*(qf32+qf42+qf42+qf52);
p2=p12+p22;

snr2(k)=((((h02^2)+(h12^2))*2*eg2/5)/n2(k));
snrdb2(k)=10*log(snr2(k));

psnr2(k)= 0.5*(1/vsnr2)*( exp( -(snr2(k)/2/vsnr2) ) );
b12(k)=p2(k)*psnr2(k);

BER2(k)=0.5*trapz(b12)*d2;

k=k+1;
end


vh03=0.1;
vh13=0.2;
ve03=0.3;
ve13=0.04;
eg3=5;
h03=10^(-19.7/10);
h13=10^(-0.9/10);
vsnr3=0.2;
```

```
d3=0.02;
x3=vh03+vh13+ve03+ve13;
k=1;
for vw3=0.1:0.03:10
vw13(k)=vw3*det([x3 0;0 x3])   %noise


y3=ve03+ve13;
vc3=(vh03*eg3*det([y3 0;0 y3]))+(vh13*eg3*det([y3 0;0 y3]))   %imperfect
channel estimator

z3=vh03+vh13;
vb3=eg3*det([z3 0;0 z3])   %ICI

n3=vw13+vc3+vb3
u13=(2*eg3/5)*((h03^2)+(h13^2));
q13(k)=sqrt(u13/n3(k));
qf13=qfunc(q13);
u23=9*u13;
q23(k)=sqrt(u23/n3(k));
qf23=qfunc(q23);
p13=0.5*(qf13+qf23);
q33(k)= u23/n3(k);
q43(k)= u13/n3(k);
q53(k)= 25*u13/n3(k);
qf33=qfunc(q33);
qf43=qfunc(q43);
qf53=qfunc(q53);
p23=0.5*(qf33+qf43+qf43+qf53);
p3=p13+p23;

snr3(k)=((((h03^2)+(h13^2))*2*eg3/5)/n3(k));
snrdb3(k)=10*log(snr3(k));

psnr3(k)= 0.5*(1/vsnr3)*( exp( -(snr3(k)/2/vsnr3) ) );
b13(k)=p3(k)*psnr3(k);

BER3(k)=0.5*trapz(b13)*d3;

k=k+1;
end

semilogy(snrdb,BER,snrdb2,BER2,snrdb3,BER3)
```

%% Code 08: Code for SNR Vs BER for 4:1, 2:1, 1:1 transmission system:

```
clc;
clear all;
close all;


N0=64;
y0=0;
for r0=1:N0-1
   m0= 1/((sin(pi*r0/N0))^2);
   y0=y0+m0;
end
v0=0.05;
g0=10^2;
a0=1;
k=1;
for e0=0:.005:.3
   d10=(v0^2)*((sinc(pi*e0))^2)*y0/(2*N0);
   d20=0.5947*((sin(pi*e0))^2)
   d30=d10+d20;
   d0=1+(g0*(a0^2)*d30);
   u0=g0*(a0^2)*(sinc(pi*e0)^2);
   snrdb0(k)=10*log10(u0/d0);
   s0(k)=(u0/d0);
   BER0(k)= 0.5*erfc(sqrt(s0(k)))
   k=k+1;
end


vh02=0.1;
vh12=0.2;

ve02=0.02;
ve12=0.04;

eg2=5;

h02=10^(-9.7/10);
h12=10^(-0.9/10);

vsnr2=0.2;

d2=0.02;
```

```
x2=vh02+vh12+ve02+ve12;

k=1;
for vw2=0.1:0.03:10
vw12(k)=vw2*det([x2 0;0 x2])   %noise

y2=ve02+ve12;
vc2=(vh02*eg2*det([y2 0;0 y2]))+(vh12*eg2*det([y2 0;0 y2]))   %imperfect
channel estimator

z2=vh02+vh12;
vb2=eg2*det([z2 0;0 z2])   %ICI

n2=vw12+vc2+vb2
u12=(2*eg2/5)*((h02^2)+(h12^2));
q12(k)=sqrt(u12/n2(k));
qf12=qfunc(q12);
u22=9*u12;
q22(k)=sqrt(u22/n2(k));
qf22=qfunc(q22);
p12=0.5*(qf12+qf22);
q32(k)= u22/n2(k);
q42(k)= u12/n2(k);
q52(k)= 25*u12/n2(k);
qf32=qfunc(q32);
qf42=qfunc(q42);
qf52=qfunc(q52);
p22=0.5*(qf32+qf42+qf42+qf52);
p2=p12+p22;

snr2(k)=((((h02^2)+(h12^2))*2*eg2/5)/n2(k));
snrdb2(k)=10*log(snr2(k));

psnr2(k)= 0.5*(1/vsnr2)*( exp( -(snr2(k)/2/vsnr2) ) );
b12(k)=p2(k)*psnr2(k);

BER2(k)=0.5*trapz(b12)*d2;

k=k+1;
end



vh04=0.1;
vh14=0.2;
```

```
vh24=0.02;
vh34=0.4;

ve04=0.1;
ve14=0.04;
ve24=0.06;
ve34=0.04;

eg4=5;
h04=10^(-9.7/10);
h14=10^(-0.9/10);
h24=10^(-8.5/10);
h34=10^(-0.5/10);

vsnr4=0.2;

d4=0.02;

x4=vh04+vh14+vh24+vh34+ve04+ve14+ve24+ve34;

k=1;
for vw4=0.1:0.03:10
vw14(k)=vw4*det([x4 0 0 0;0 x4 0 0 ;0 0 x4 0;0 0 0 x4])   %noise

y4=ve04+ve14+ve24+ve34;
vc4=(vh04*eg4*det([y4 0 0 0;0 y4 0 0 ;0 0 y4 0;0 0 0 y4]))+(vh14*eg4*det([y4 0 0
0;0 y4 0 0 ;0 0 y4 0;0 0 0 y4]))+(vh24*eg4*det([y4 0 0 0;0 y4 0 0 ;0 0 y4 0;0 0 0
y4]))+(vh34*eg4*det([y4 0 0 0;0 y4 0 0 ;0 0 y4 0;0 0 0 y4]));   %imperfect channel
estimator


z4=vh04+vh14+vh24+vh34;
vb4=eg4*det([z4 0 0 0;0 z4 0 0;0 0 z4 0;0 0 0 z4])   %ICI

n4=vw14+vc4+vb4
u14=(2*eg4/5)*((h04^2)+(h14^2)+(h24^2)+(h34^2));
q14(k)=sqrt(u14/n4(k));
qf14=qfunc(q14);
u24=9*u14;
q24(k)=sqrt(u24/n4(k));
qf24=qfunc(q24);
p14=0.5*(qf14+qf24);
q34(k)= u24/n4(k);
q44(k)= u14/n4(k);
q54(k)= 25*u14/n4(k);
qf34=qfunc(q34);
```

```
qf44=qfunc(q44);
qf54=qfunc(q54);
p24=0.5*(qf34+qf44+qf44+qf54);
p4=p14+p24;

snr4(k)=((((h04^2)+(h14^2)+(h24^2)+(h34^3))*2*eg4/5)/n4(k));
snrdb4(k)=10*log(snr4(k));

psnr4(k)= 0.5*(1/vsnr4)*( exp( -(snr4(k)/2/vsnr4) ) );
b14(k)=p4(k)*psnr4(k);

BER4(k)=0.5*trapz(b14)*d4;

k=k+1;
end

semilogy(snrdb0,BER0,snrdb2,BER2,snrdb4,BER4)
```

%% Code 09: SNR Vs BER graph for different value of noises
%% (Tx=4 ; Rx=1):

```
clc;
clear all;
close all;

vh041=0.1;
vh141=0.2;
vh241=0.02;
vh341=0.4;


ve041=0.1;
ve141=0.04;
ve241=0.06;
ve341=0.04;

eg41=5;
h041=10^(-9.7/10);
h141=10^(-0.9/10);
h241=10^(-8.5/10);
h341=10^(-0.5/10);
vsnr41=0.2;
d41=0.02;
x41=vh041+vh141+vh241+vh341+ve041+ve141+ve241+ve341;
k=1;
```

```
for vw41=0.1:0.03:10
vw141(k)=vw41*det([x41 0 0 0;0 x41 0 0 ;0 0 x41 0;0 0 0 x41])   %noise


y41=ve041+ve141+ve241+ve341;
vc41=(vh041*eg41*det([y41 0 0 0;0 y41 0 0 ;0 0 y41 0;0 0 0
y41]))+(vh141*eg41*det([y41 0 0 0;0 y41 0 0 ;0 0 y41 0;0 0 0
y41]))+(vh241*eg41*det([y41 0 0 0;0 y41 0 0 ;0 0 y41 0;0 0 0
y41]))+(vh341*eg41*det([y41 0 0 0;0 y41 0 0 ;0 0 y41 0;0 0 0 y41]));   %imperfect
channel estimator

z41=vh041+vh141+vh241+vh341;
vb41=eg41*det([z41 0 0 0;0 z41 0 0;0 0 z41 0;0 0 0 z41])   %ICI

n41=vw141+vc41+vb41
u141=(2*eg41/5)*((h041^2)+(h141^2)+(h241^2)+(h341^2));
q141(k)=sqrt(u141/n41(k));
qf141=qfunc(q141);
u241=9*u141;
q241(k)=sqrt(u241/n41(k));
qf241=qfunc(q241);
p141=0.5*(qf141+qf241);
q341(k)= u241/n41(k);
q441(k)= u141/n41(k);
q541(k)= 25*u141/n41(k);
qf341=qfunc(q341);
qf441=qfunc(q441);
qf541=qfunc(q541);
p241=0.5*(qf341+qf441+qf441+qf541);
p41=p141+p241;

snr41(k)=((((h041^2)+(h141^2)+(h241^2)+(h341^3))*2*eg41/5)/n41(k));
snrdb41(k)=10*log(snr41(k));

psnr41(k)= 0.5*(1/vsnr41)*( exp( -(snr41(k)/2/vsnr41) ) );
b141(k)=p41(k)*psnr41(k);

BER41(k)=0.5*trapz(b141)*d41;

k=k+1;
end


vh042=0.04;
vh142=0.05;
vh242=0.7;
```

```
vh342=0.1;


ve042=0.1;
ve142=0.04;
ve242=0.06;
ve342=0.04;

eg42=5;
h042=10^(-9.7/10);
h142=10^(-0.9/10);
h242=10^(-8.5/10);
h342=10^(-0.5/10);
vsnr42=0.2;
d42=0.02;
x42=vh042+vh142+vh242+vh342+ve042+ve142+ve242+ve342;
k=1;
for vw42=0.1:0.03:10
vw142(k)=vw42*det([x42 0 0 0;0 x42 0 0 ;0 0 x42 0;0 0 0 x42])   %noise


y42=ve042+ve142+ve242+ve342;
vc42=(vh042*eg42*det([y42 0 0 0;0 y42 0 0 ;0 0 y42 0;0 0 0
y42]))+(vh142*eg42*det([y42 0 0 0;0 y42 0 0 ;0 0 y42 0;0 0 0
y42]))+(vh242*eg42*det([y42 0 0 0;0 y42 0 0 ;0 0 y42 0;0 0 0
y42]))+(vh342*eg42*det([y42 0 0 0;0 y42 0 0 ;0 0 y42 0;0 0 0 y42]));   %imperfect
channel estimator

z42=vh042+vh142+vh242+vh342;
vb42=eg42*det([z42 0 0 0;0 z42 0 0;0 0 z42 0;0 0 0 z42])   %ICI

n42=vw142+vc42+vb42
u142=(2*eg42/5)*((h042^2)+(h142^2)+(h242^2)+(h342^2));
q142(k)=sqrt(u142/n42(k));
qf142=qfunc(q142);
u242=9*u142;
q242(k)=sqrt(u242/n42(k));
qf242=qfunc(q242);
p142=0.5*(qf142+qf242);
q342(k)= u242/n42(k);
q442(k)= u142/n42(k);
q542(k)= 25*u142/n42(k);
qf342=qfunc(q342);
qf442=qfunc(q442);
qf542=qfunc(q542);
p242=0.5*(qf342+qf442+qf442+qf542);
```

```
p42=p142+p242;

snr42(k)=((((h042^2)+(h142^2)+(h242^2)+(h342^3))*2*eg42/5)/n42(k));
snrdb42(k)=10*log(snr42(k));

psnr42(k)= 0.5*(1/vsnr42)*( exp( -(snr42(k)/2/vsnr42) ) );
b142(k)=p42(k)*psnr42(k);

BER42(k)=0.5*trapz(b142)*d42;

k=k+1;
end


vh043=0.04;
vh143=0.05;
vh243=0.8;
vh343=0.1;


ve043=0.1;
ve143=0.04;
ve243=0.06;
ve343=0.04;

eg43=5;
h043=10^(-9.7/10);
h143=10^(-0.9/10);
h243=10^(-8.5/10);
h343=10^(-0.5/10);
vsnr43=0.2;
d43=0.02;
x43=vh043+vh143+vh243+vh343+ve043+ve143+ve243+ve343;
k=1;
for vw43=0.1:0.03:10
vw143(k)=vw43*det([x43 0 0 0;0 x43 0 0 ;0 0 x43 0;0 0 0 x43])   %noise


y43=ve043+ve143+ve243+ve343;
vc43=(vh043*eg43*det([y43 0 0 0;0 y43 0 0 ;0 0 y43 0;0 0 0
y43]))+(vh143*eg43*det([y43 0 0 0;0 y43 0 0 ;0 0 y43 0;0 0 0
y43]))+(vh243*eg43*det([y43 0 0 0;0 y43 0 0 ;0 0 y43 0;0 0 0
y43]))+(vh343*eg43*det([y43 0 0 0;0 y43 0 0 ;0 0 y43 0;0 0 0 y43]));   %imperfect
channel estimator

z43=vh043+vh143+vh243+vh343;
```

```
vb43=eg43*det([z43 0 0 0;0 z43 0 0;0 0 z43 0;0 0 0 z43])   %ICI

n43=vw143+vc43+vb43
u143=(2*eg43/5)*((h043^2)+(h143^2)+(h243^2)+(h343^2));
q143(k)=sqrt(u143/n43(k));
qf143=qfunc(q143);
u243=9*u143;
q243(k)=sqrt(u243/n43(k));
qf243=qfunc(q243);
p143=0.5*(qf143+qf243);
q343(k)= u243/n43(k);
q443(k)= u143/n43(k);
q543(k)= 25*u143/n43(k);
qf343=qfunc(q343);
qf443=qfunc(q443);
qf543=qfunc(q543);
p243=0.5*(qf343+qf443+qf443+qf543);
p43=p143+p243;

snr43(k)=((((h043^2)+(h143^2)+(h243^2)+(h343^3))*2*eg43/5)/n43(k));
snrdb43(k)=10*log(snr43(k));

psnr43(k)= 0.5*(1/vsnr43)*( exp( -(snr43(k)/2/vsnr43) ) );
b143(k)=p43(k)*psnr43(k);

BER43(k)=0.5*trapz(b143)*d43;

k=k+1;
end



semilogy(snrdb41,BER41,snrdb42,BER42,snrdb43,BER43)
```

%% Code 10: Code for SNR Vs BER for 6:1, 4:1, 2:1, 1:1 transmission
%% System:

```
clc;
clear all;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%Without Diversity (Tx=1;
Rx=1)%%%%%%%%%%%%%%%%%%%%%%%%%

N0=64;
y0=0;
for r0=1:N0-1
   m0= 1/((sin(pi*r0/N0))^2);
   y0=y0+m0;
end
v0=0.05;
g0=10^2;
a0=1;
k=1;
for e0=0:.005:.3
   d10=(v0^2)*((sinc(pi*e0))^2)*y0/(2*N0);
   d20=0.5947*((sin(pi*e0))^2)
   d30=d10+d20;
   d0=1+(g0*(a0^2)*d30);
   u0=g0*(a0^2)*(sinc(pi*e0)^2);
   snrdb0(k)=10*log10(u0/d0);
   s0(k)=(u0/d0);
   BER0(k)= 0.5*erfc(sqrt(s0(k)))
   k=k+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%Using Diversity (Tx=2;
Rx=1)%%%%%%%%%%%%%%%%%%%%%%%%%

vh02=0.1;
vh12=0.2;

ve02=0.02;
ve12=0.04;

eg2=5;

h02=10^(-9.7/10);
```

```
h12=10^(-0.9/10);

vsnr2=0.2;
d2=0.02;

x2=vh02+vh12+ve02+ve12;

k=1;
for vw2=0.1:0.03:10
vw12(k)=vw2*det([x2 0;0 x2])   %noise

y2=ve02+ve12;
vc2=(vh02*eg2*det([y2 0;0 y2]))+(vh12*eg2*det([y2 0;0 y2]))   %imperfect
channel estimator

z2=vh02+vh12;
vb2=eg2*det([z2 0;0 z2])   %ICI

n2=vw12+vc2+vb2

u12=(2*eg2/5)*((h02^2)+(h12^2));
q12(k)=sqrt(u12/n2(k));
qf12=qfunc(q12);
u22=9*u12;
q22(k)=sqrt(u22/n2(k));
qf22=qfunc(q22);
p12=0.5*(qf12+qf22);
q32(k)= u22/n2(k);
q42(k)= u12/n2(k);
q52(k)= 25*u12/n2(k);
qf32=qfunc(q32);
qf42=qfunc(q42);
qf52=qfunc(q52);
p22=0.5*(qf32+qf42+qf42+qf52);
p2=p12+p22;

snr2(k)=((((h02^2)+(h12^2))*2*eg2/5)/n2(k));
snrdb2(k)=10*log(snr2(k));

psnr2(k)= 0.5*(1/vsnr2)*( exp( -(snr2(k)/2/vsnr2) ) );
b12(k)=p2(k)*psnr2(k);

BER2(k)=0.5*trapz(b12)*d2;

k=k+1;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%Using Diversity (Tx=4;
Rx=1)%%%%%%%%%%%%%%%%%%%%%%%%%%%%

vh04=0.1;
vh14=0.2;
vh24=0.02;
vh34=0.4;


ve04=0.1;
ve14=0.04;
ve24=0.06;
ve34=0.04;

eg4=5;
h04=10^(-9.7/10);
h14=10^(-0.9/10);
h24=10^(-8.5/10);
h34=10^(-0.5/10);
vsnr4=0.2;
d4=0.02;
x4=vh04+vh14+vh24+vh34+ve04+ve14+ve24+ve34;
k=1;
for vw4=0.1:0.03:10
vw14(k)=vw4*det([x4 0 0 0;0 x4 0 0 ;0 0 x4 0;0 0 0 x4])   %noise


y4=ve04+ve14+ve24+ve34;
vc4=(vh04*eg4*det([y4 0 0 0;0 y4 0 0 ;0 0 y4 0;0 0 0 y4]))+(vh14*eg4*det([y4 0 0
0;0 y4 0 0 ;0 0 y4 0;0 0 0 y4]))+(vh24*eg4*det([y4 0 0 0;0 y4 0 0 ;0 0 y4 0;0 0 0
y4]))+(vh34*eg4*det([y4 0 0 0;0 y4 0 0 ;0 0 y4 0;0 0 0 y4]));   %imperfect channel
estimator


z4=vh04+vh14+vh24+vh34;
vb4=eg4*det([z4 0 0 0;0 z4 0 0;0 0 z4 0;0 0 0 z4])   %ICI

n4=vw14+vc4+vb4
u14=(2*eg4/5)*((h04^2)+(h14^2)+(h24^2)+(h34^2));
q14(k)=sqrt(u14/n4(k));
qf14=qfunc(q14);
u24=9*u14;
q24(k)=sqrt(u24/n4(k));
qf24=qfunc(q24);
p14=0.5*(qf14+qf24);
```

```
q34(k)= u24/n4(k);
q44(k)= u14/n4(k);
q54(k)= 25*u14/n4(k);
qf34=qfunc(q34);
qf44=qfunc(q44);
qf54=qfunc(q54);
p24=0.5*(qf34+qf44+qf44+qf54);
p4=p14+p24;

snr4(k)=((((h04^2)+(h14^2)+(h24^2)+(h34^3))*2*eg4/5)/n4(k));
snrdb4(k)=10*log(snr4(k));

psnr4(k)= 0.5*(1/vsnr4)*( exp( -(snr4(k)/2/vsnr4) ) );
b14(k)=p4(k)*psnr4(k);

BER4(k)=0.5*trapz(b14)*d4;

k=k+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%Using Diversity (Tx=6;
Rx=1)%%%%%%%%%%%%%%%%%%%%%%%%%%%%

vh06=0.1;
vh16=0.2;
vh26=0.02;
vh36=0.4;
vh46=0.01;
vh56=0.03;

ve06=0.01;
ve16=0.04;
ve26=0.06;
ve36=0.04;
ve46=0.02;
ve56=0.03;

eg6=5;

h06=10^(-9.7/10);
h16=10^(-0.9/10);
h26=10^(-8.5/10);
h36=10^(-0.5/10);
h46=10^(-0.8/10);
h56=10^(-7.5/10);
```

```
vsnr6=0.2;
d6=0.02;

x6=vh06+vh16+vh26+vh36+vh46+vh46+ve06+ve16+ve26+ve36+ve46+ve56;

k=1;

for vw6=0.1:0.03:10
vw16(k)=vw6*det([x6 0 0 0 0 0;0 x6 0 0 0 0 ;0 0 x6 0 0 0 ;0 0 0 x6 0 0;0 0 0 0 x6
0;0 0 0 0 0 x6])     %noise

y6=ve06+ve16+ve26+ve36;
vc6=((vh06*eg6*det([y6 0 0 0 0 0;0 y6 0 0 0 0 ;0 0 y6 0 0 0;0 0 0 y6 0 0;0 0 0 0
y6 0;0 0 0 0 0 y6]))+ (vh16*eg6*det([y6 0 0 0 0 0;0 y6 0 0 0 0 ;0 0 y6 0 0 0;0 0 0
y6 0 0;0 0 0 0 y6 0;0 0 0 0 0 y6]))+(vh26*eg6*det([y6 0 0 0 0 0;0 y6 0 0 0 0 ;0 0
y6 0 0 0;0 0 0 y6 0 0;0 0 0 0 y6 0;0 0 0 0 0 y6]))+(vh36*eg6*det([y6 0 0 0 0 0;0 y6
0 0 0 0 ;0 0 y6 0 0 0;0 0 0 y6 0 0;0 0 0 0 y6 0;0 0 0 0 0 y6]))+(vh46*eg6*det([y6 0
0 0 0 0;0 y6 0 0 0 0 ;0 0 y6 0 0 0;0 0 0 y6 0 0;0 0 0 0 y6 0;0 0 0 0 0
y6]))+(vh56*eg6*det([y6 0 0 0 0 0;0 y6 0 0 0 0 ;0 0 y6 0 0 0;0 0 0 y6 0 0;0 0 0 0
y6 0;0 0 0 0 0 y6]))); %imperfect channel estimator

z6=vh06+vh16+vh26+vh36+vh46+vh56;
vb6=eg6*det([z6 0 0 0 0 0;0 z6 0 0 0 0 ;0 0 z6 0 0 0;0 0 0 z6 0 0;0 0 0 0 z6 0;0 0
0 0 0 z6])   %ICI

n6=vw16+vc6+vb6

u16=(2*eg6/5)*((h06^2)+(h16^2)+(h26^2)+(h36^2)+(h46^2)+(h56^2));
q16(k)=sqrt(u16/n6(k));
qf16=qfunc(q16);
u26=9*u16;
q26(k)=sqrt(u26/n6(k));
qf26=qfunc(q26);
p16=0.5*(qf16+qf26);
q36(k)= u26/n6(k);
q46(k)= u16/n6(k);
q56(k)= 25*u16/n6(k);
qf36=qfunc(q36);
qf46=qfunc(q46);
qf56=qfunc(q56);
p26=0.5*(qf36+qf46+qf46+qf56);
p6=p16+p26;

snr6(k)=((((h06^2)+(h16^2)+(h26^2)+(h36^3)+(h46^2)+(h56^2)*2*eg6/5)/n6(k)));
snrdb6(k)=10*log(snr6(k));
```

```
psnr6(k)= 0.5*(1/vsnr6)*( exp( -(snr6(k)/2/vsnr6) ) );
b16(k)=p6(k)*psnr6(k);

BER6(k)=0.5*trapz(b16)*d6;

k=k+1;
end

semilogy(snrdb0,BER0,snrdb2,BER2,snrdb4,BER4,snrdb6,BER6)
```

%% Code 11: Code for SNR Vs BER graph for different value of noises
%% (Tx=6 ; Rx=1):

```
clc;
clear all;
close all;


vh061=0.1;
vh161=0.2;
vh261=0.02;
vh361=0.4;
vh461=0.01;
vh561=0.03;

ve061=0.01;
ve161=0.04;
ve261=0.06;
ve361=0.04;
ve461=0.02;
ve561=0.03;

eg61=5;

h061=10^(-9.7/10);
h161=10^(-0.9/10);
h261=10^(-8.5/10);
h361=10^(-0.5/10);
h461=10^(-0.8/10);
h561=10^(-7.5/10);

vsnr61=0.2;
d61=0.02;
```

```
x61=vh061+vh161+vh261+vh361+vh461+vh461+ve061+ve161+ve261+ve361+v
e461+ve561;

k=1;

for vw61=0.1:0.03:10
vw161(k)=vw61*det([x61 0 0 0 0 0;0 x61 0 0 0 0 ;0 0 x61 0 0 0 ;0 0 0 x61 0 0;0 0
0 0 x61 0;0 0 0 0 0 x61])     %noise

y61=ve061+ve161+ve261+ve361;
vc61=((vh061*eg61*det([y61 0 0 0 0 0;0 y61 0 0 0 0 ;0 0 y61 0 0 0;0 0 0 y61 0
0;0 0 0 0 y61 0;0 0 0 0 0 y61]))+ (vh161*eg61*det([y61 0 0 0 0 0;0 y61 0 0 0 0 ;0
0 y61 0 0 0;0 0 0 y61 0 0;0 0 0 0 y61 0;0 0 0 0 0 y61]))+(vh261*eg61*det([y61 0 0
0 0 0;0 y61 0 0 0 0 ;0 0 y61 0 0 0;0 0 0 y61 0 0;0 0 0 0 y61 0;0 0 0 0 0
y61]))+(vh361*eg61*det([y61 0 0 0 0 0;0 y61 0 0 0 0 ;0 0 y61 0 0 0;0 0 0 y61 0
0;0 0 0 0 y61 0;0 0 0 0 0 y61]))+(vh461*eg61*det([y61 0 0 0 0 0;0 y61 0 0 0 0 ;0
0 y61 0 0 0;0 0 0 y61 0 0;0 0 0 0 y61 0;0 0 0 0 0 y61]))+(vh561*eg61*det([y61 0 0
0 0 0;0 y61 0 0 0 0 ;0 0 y61 0 0 0;0 0 0 y61 0 0;0 0 0 0 y61 0;0 0 0 0 0 y61])));
%imperfect channel estimator

z61=vh061+vh161+vh261+vh361+vh461+vh561;
vb61=eg61*det([z61 0 0 0 0 0;0 z61 0 0 0 0 ;0 0 z61 0 0 0;0 0 0 z61 0 0;0 0 0 0
z61 0;0 0 0 0 0 z61])   %ICI

n61=vw161+vc61+vb61

u161=(2*eg61/5)*((h061^2)+(h161^2)+(h261^2)+(h361^2)+(h461^2)+(h561^2));
q161(k)=sqrt(u161/n61(k));
qf161=qfunc(q161);
u261=9*u161;
q261(k)=sqrt(u261/n61(k));
qf261=qfunc(q261);
p161=0.5*(qf161+qf261);
q361(k)= u261/n61(k);
q461(k)= u161/n61(k);
q561(k)= 25*u161/n61(k);
qf361=qfunc(q361);
qf461=qfunc(q461);
qf561=qfunc(q561);
p261=0.5*(qf361+qf461+qf461+qf561);
p61=p161+p261;

snr61(k)=((((h061^2)+(h161^2)+(h261^2)+(h361^3)+(h461^2)+(h561^2)*2*eg61/
5)/n61(k)));
snrdb61(k)=10*log(snr61(k));
```

```
psnr61(k)= 0.5*(1/vsnr61)*( exp( -(snr61(k)/2/vsnr61) ) );
b161(k)=p61(k)*psnr61(k);

BER61(k)=0.5*trapz(b161)*d61;

k=k+1;
end

vh062=0.1;
vh162=0.2;
vh262=0.02;
vh362=0.4;
vh462=0.01;
vh562=0.03;

ve062=0.02;
ve162=0.05;
ve262=0.07;
ve362=0.05;
ve462=0.03;
ve562=0.04;

eg62=5;

h062=10^(-9.7/10);
h162=10^(-0.9/10);
h262=10^(-8.5/10);
h362=10^(-0.5/10);
h462=10^(-0.8/10);
h562=10^(-7.5/10);

vsnr62=0.2;
d62=0.02;

x62=vh062+vh162+vh262+vh362+vh462+vh462+ve062+ve162+ve262+ve362+v
e462+ve562;

k=1;

for vw62=0.1:0.03:10
vw162(k)=vw62*det([x62 0 0 0 0 0;0 x62 0 0 0 0 ;0 0 x62 0 0 0 ;0 0 0 x62 0 0;0 0
0 0 x62 0;0 0 0 0 0 x62])     %noise

y62=ve062+ve162+ve262+ve362;
vc62=((vh062*eg62*det([y62 0 0 0 0 0;0 y62 0 0 0 0 ;0 0 y62 0 0 0;0 0 0 y62 0
0;0 0 0 0 y62 0;0 0 0 0 0 y62]))+ (vh162*eg62*det([y62 0 0 0 0 0;0 y62 0 0 0 0 ;0
```

```
0 y62 0 0 0;0 0 0 y62 0 0;0 0 0 0 y62 0;0 0 0 0 0 y62]))+(vh262*eg62*det([y62 0 0
0 0 0;0 y62 0 0 0 0 ;0 0 y62 0 0 0;0 0 0 y62 0 0;0 0 0 0 y62 0;0 0 0 0 0
y62]))+(vh362*eg62*det([y62 0 0 0 0 0;0 y62 0 0 0 0 ;0 0 y62 0 0 0;0 0 0 y62 0
0;0 0 0 0 y62 0;0 0 0 0 0 y62]))+(vh462*eg62*det([y62 0 0 0 0 0;0 y62 0 0 0 0 ;0
0 y62 0 0 0;0 0 0 y62 0 0;0 0 0 0 y62 0;0 0 0 0 0 y62]))+(vh562*eg62*det([y62 0 0
0 0 0;0 y62 0 0 0 0 ;0 0 y62 0 0 0;0 0 0 y62 0 0;0 0 0 0 y62 0;0 0 0 0 0 y62])));
%imperfect channel estimator

z62=vh062+vh162+vh262+vh362+vh462+vh562;
vb62=eg62*det([z62 0 0 0 0 0;0 z62 0 0 0 0 ;0 0 z62 0 0 0;0 0 0 z62 0 0;0 0 0 0
z62 0;0 0 0 0 0 z62])   %ICI

n62=vw162+vc62+vb62

u162=(2*eg62/5)*((h062^2)+(h162^2)+(h262^2)+(h362^2)+(h462^2)+(h562^2));
q162(k)=sqrt(u162/n62(k));
qf162=qfunc(q162);
u262=9*u162;
q262(k)=sqrt(u262/n62(k));
qf262=qfunc(q262);
p162=0.5*(qf162+qf262);
q362(k)= u262/n62(k);
q462(k)= u162/n62(k);
q562(k)= 25*u162/n62(k);
qf362=qfunc(q362);
qf462=qfunc(q462);
qf562=qfunc(q562);
p262=0.5*(qf362+qf462+qf462+qf562);
p62=p162+p262;

snr62(k)=((((h062^2)+(h162^2)+(h262^2)+(h362^3)+(h462^2)+(h562^2)*2*eg62/
5)/n62(k)));
snrdb62(k)=10*log(snr62(k));

psnr62(k)= 0.5*(1/vsnr62)*( exp( -(snr62(k)/2/vsnr62) ) );
b162(k)=p62(k)*psnr62(k);

BER62(k)=0.5*trapz(b162)*d62;

k=k+1;
end


vh063=0.1;
vh163=0.2;
vh263=0.02;
```

```
vh363=0.4;
vh463=0.01;
vh563=0.03;

ve063=0.03;
ve163=0.06;
ve263=0.08;
ve363=0.06;
ve463=0.04;
ve563=0.05;

eg63=5;

h063=10^(-9.7/10);
h163=10^(-0.9/10);
h263=10^(-8.5/10);
h363=10^(-0.5/10);
h463=10^(-0.8/10);
h563=10^(-7.5/10);

vsnr63=0.2;
d63=0.02;

x63=vh063+vh163+vh263+vh363+vh463+vh463+ve063+ve163+ve263+ve363+v
e463+ve563;

k=1;

for vw63=0.1:0.03:10
vw163(k)=vw63*det([x63 0 0 0 0 0;0 x63 0 0 0 0 ;0 0 x63 0 0 0 ;0 0 0 x63 0 0;0 0
0 0 x63 0;0 0 0 0 0 x63])     %noise

y63=ve063+ve163+ve263+ve363;
vc63=((vh063*eg63*det([y63 0 0 0 0 0;0 y63 0 0 0 0 ;0 0 y63 0 0 0;0 0 0 y63 0
0;0 0 0 0 y63 0;0 0 0 0 0 y63]))+ (vh163*eg63*det([y63 0 0 0 0 0;0 y63 0 0 0 0 ;0
0 y63 0 0 0;0 0 0 y63 0 0;0 0 0 0 y63 0;0 0 0 0 0 y63]))+(vh263*eg63*det([y63 0 0
0 0 0;0 y63 0 0 0 0 ;0 0 y63 0 0 0;0 0 0 y63 0 0;0 0 0 0 y63 0;0 0 0 0 0
y63]))+(vh363*eg63*det([y63 0 0 0 0 0;0 y63 0 0 0 0 ;0 0 y63 0 0 0;0 0 0 y63 0
0;0 0 0 0 y63 0;0 0 0 0 0 y63]))+(vh463*eg63*det([y63 0 0 0 0 0;0 y63 0 0 0 0 ;0
0 y63 0 0 0;0 0 0 y63 0 0;0 0 0 0 y63 0;0 0 0 0 0 y63]))+(vh563*eg63*det([y63 0 0
0 0 0;0 y63 0 0 0 0 ;0 0 y63 0 0 0;0 0 0 y63 0 0;0 0 0 0 y63 0;0 0 0 0 0 y63])));
%imperfect channel estimator

z63=vh063+vh163+vh263+vh363+vh463+vh563;
vb63=eg63*det([z63 0 0 0 0 0;0 z63 0 0 0 0 ;0 0 z63 0 0 0;0 0 0 z63 0 0;0 0 0 0
z63 0;0 0 0 0 0 z63])   %ICI
```

```
n63=vw163+vc63+vb63

u163=(2*eg63/5)*((h063^2)+(h163^2)+(h263^2)+(h363^2)+(h463^2)+(h563^2));
q163(k)=sqrt(u163/n63(k));
qf163=qfunc(q163);
u263=9*u163;
q263(k)=sqrt(u263/n63(k));
qf263=qfunc(q263);
p163=0.5*(qf163+qf263);
q363(k)= u263/n63(k);
q463(k)= u163/n63(k);
q563(k)= 25*u163/n63(k);
qf363=qfunc(q363);
qf463=qfunc(q463);
qf563=qfunc(q563);
p263=0.5*(qf363+qf463+qf463+qf563);
p63=p163+p263;

snr63(k)=((((h063^2)+(h163^2)+(h263^2)+(h363^3)+(h463^2)+(h563^2)*2*eg63/
5)/n63(k)));
snrdb63(k)=10*log(snr63(k));

psnr63(k)= 0.5*(1/vsnr63)*( exp( -(snr63(k)/2/vsnr63) ) );
b163(k)=p63(k)*psnr63(k);

BER63(k)=0.5*trapz(b163)*d63;

k=k+1;
end

semilogy(snrdb61,BER61,snrdb62,BER62,snrdb63,BER63)
```

%% Code 12: Code for Rx Sensitivity for BER 1e-3:

```
clc;
close all;

txn=[6 4 2 1];

snrdb1=[-1.24 2.5 4.5 6.5];

snrdb2=[-2 1.25 4 6.5];

snrdb3=[-4 -1 1 6.5];
plot(txn,snrdb1,txn,snrdb2,txn,snrdb3)
```