BUSSGANG ALGORITHMS COMPARISON FOR BLIND EQUALIZATION OF CHANNEL

An MS Final Year Dissertation *by*

Muhammad Salman
125-FET/MSEE/F07

Supervised by
Dr. Aqdas Naveed Malik

Department of Electronic Engineering
Faculty of Engineering and Technology

INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD

March 2008

14-07-2010

MS

621-3822
SAB

T0445E2007DEEMS 1-signal processing- Digital techniques.
2- Digital Communication

MS

Accession No 4452

SAB

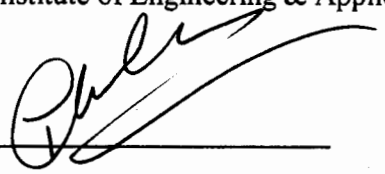
DATA ENTERED
08/02/2012

D.E
AC
3.8.11

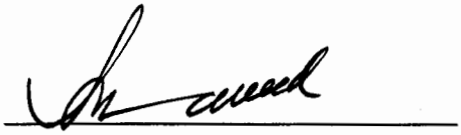
Certificate

It is certified that the research work contained in this thesis by **Muhammad Salman [125-FET/MSEE/F07]** has been carried out under my supervision. In my opinion it is fully adequate, in scope and quality, as a thesis for the degree of MS(TE) in Telecommunication Engineering awarded by International Islamic University, Islamabad.

External Examiner
Dr. Abdul Jalil
Department of Electrical Engineering,
Faculty of Engineering
Pakistan Institute of Engineering & Applied Sciences, Islamabad



Internal Examiner
Dr. Tanveer Ahmed Cheema
Assistant Professor
Department of Electronic Engineering,
Faculty of Engineering & Technology
International Islamic University, Islamabad

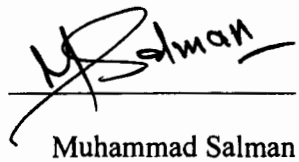


Supervisor
Dr. Aqdas Naveed Malik
Assistant Professor
Department of Electronic Engineering,
Faculty of Engineering & Technology,
International Islamic University, Islamabad

Declaration

I hereby, declare that this thesis, neither as a whole nor as a part thereof has been copied out from any source. It is further declare that I have developed this thesis and the accompanied report entirely on the basis of my personal effort made under the guidance of our teachers.

If any part of this report to be copied out or found to be reported, I shall standby the consequences, no portion pf this work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.



Muhammad Salman

125-FET/MSEE/F07

A dissertation submitted to the Department of Electronic Engineering

International Islamic University, Islamabad in partial fulfillment of the

requirements for the award of the degree of

Master of Science in Telecommunication Engineering

To my parents,

loved ones, and faculty members

Acknowledgement

First, I humbly praise and thank AlMIGHTY ALLAH, The compassionate and The merciful, Who gave health, thoughts, affectionate parents, talented teachers, helping friends and opportunity to contribute to the vast pool of knowledge. Peace and prayers for His Prophet HAZRAT MUHAMMAD (S.A.W.) whose incomparable life is the glorious model for humanity.

I would like to express my gratitude to Dr. Aqdas Naveed Malik for constant attention to my thesis, helpful suggestions, remarks, discussions and affectionate behavior helped me a lot in the completion of my thesis.

I thank Professor Dr. Ijaz Mansoor Qureshi who provided a knowledge base from which I benefited immensely and sound advice, criticism and an incredible range of technical and administrative support.

I would like to thank my family and loved ones for providing emotional support, guidance and prayers that enabled me to decide about my future.

(Muhammad Salman)

Abstract

Digital communication experience intersymbol interference (ISI), which is a major problem in high speed data transfer. Equalization is performed by minimizing a cost function to reduce ISI. Channel equalization is of two type adaptive and blind equalization. In adaptive method training sequence is send in advance to equalize the channel, which is a drawback. Blind equalization problem estimates the transmitted data and channel impulse response from the structure and statistics of the data transferred. Blind equalization can correct phase distortion but have poor convergence. Bussgang methods are one of the suggested algorithms for the solution of blind equalization problems. In my work comparison of bussgang algorithms are done and computer simulation shows that multimodulus algorithm implemented with received antenna array give best results.

Comparison of multimodulus algorithms for blind channel equalizations of complex communication channels derived by solving the constrained optimization with relaxation has been considered. The Received equalized outputs, symbol error rate (SER), mean square error (MSE) and normalized root mean square intersymbol interference (NRMS-ISI) optimizations are analyzed. It is shown with computer simulation that superior performance is obtained by using receive antenna array for multimodulus algorithm over others.

Comparison of Modified CMA presented by Oh and Chin [10], New Multimodulus Blind Equalization Algorithm with Relaxation derived by Shafat Abrar and Ismail Shah [11] and Shafat's algorithm is implemented with received antenna array and the proposed algorithm yields stable convergence with consistent lowering in ISI floor without affecting the convergence speed.

Project in Brief

Project Title:	**Bussgang Algorithms Comparison for Blind Equalization of Channel**
Undertaken by:	Muhammad Salman 125-FET/MSEE/F07
Supervised by:	Dr. Aqdas Naveed Malik
Date Started:	September 2007
Date Completed:	March 2008
Tool Used:	Matlab v7

Table of Contents

List of Figures

List of Acronyms

AWGN	Additive White Gaussian Noise
CMA	Constant Modulus Algorithm
ISI	Inter Symbol Interference
M-ary PAM	Multilevel Pulse-Amplitude Modulation
MCMA	Modified Constant Modulus Algorithm
MMA	Multimodulus Algorithms
MSE	Mean Square Error
NMBEAR	New Multimodulus Blind Equalization Algorithm with Relaxation
NRMS IS1	Normalized Root Mean Square Inter Symbol Interference
PAM	Pulse-Amplitude Modulation
QAM	Quadrature-Amplitude Modulation
SER	Symbol Error Rate
SNR	Signal-to-Noise Ratio

Chapter 1

Introduction to Bussgang Algorithms

1.1. Adaptive Equalization

Digital communication required high speed data transmission over band limited channels. These channels suffer from intersymbol interference (ISI) and to cancel the effect of ISI adaptive algorithms are used. The main advantage includes faster convergence where as, initial transmitted data for equalization is a drawback. Fig 1.1 shows channel equalization model where a_k is the transmitted symbol and $\hat{a}_k$ are the estimated symbols. The factors that limit transmission of digital data are intersymbol interference (ISI) and noise.

Digital communications requires that the transmission system should use the channel bandwidth efficiently. System is designed to adjust highest possible data transmission and its reliability is calculated using symbol error probability and error rate. The factors that limit transmission of digital data are intersymbol interference (ISI) and noise. Transmitted filter, channel, receive filter and causes ISI whereas noise is generated by the receiver. Dispersion in the transmitted filter causes ISI and it is minimized while designing high data rate transmission system.

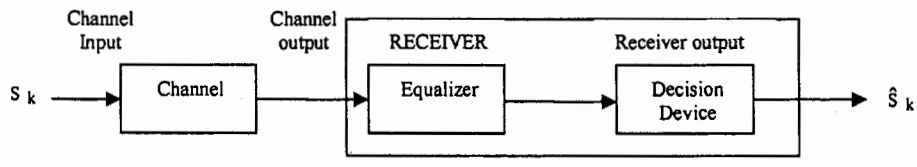


Fig 1.1 Channel Equalization model

Pulse amplitude modulation (PAM) system equivalent block diagram is shown in Fig 1.2. Binary symbols consisting of symbols 0 or 1 are the signals transmitted. The signal reaches the receiver while passing from the pulse generator, transmitter and medium. The output denoted as received filter in the block diagram can be represented as u (k), the sampling is performed in synchronism with the pulse generator in the transmitter. A threshold is used to decide the output, if the output is greater that the threshold set the decision is for the symbol 1 otherwise 0 was sent.

Let a_k be a scaling factor defined by,

$$a_k = \begin{cases} +1 \text{ if the input consist of symbol } 1 \\ -1 \text{ if the input consist of symbol } 0 \end{cases}$$

Then in the absence of noise, we may express

$$\begin{aligned} u(k) &= \sum_n a_n + p(k - n) \\ &= a_k * p(0) + \sum_n a_n + p(k - n) \qquad (n \neq 0) \end{aligned} \tag{1.1}$$

The impulse response of transmitter filter, the transmission medium, and the receiver filter is denoted by $p(n)$. Eq. 1.1 right hand side denoted the desired symbol and the remaining is the ISI cause by the channel. If the ISI is not catered while designing the system, the received signal will have errors and correct results cannot be achieved.

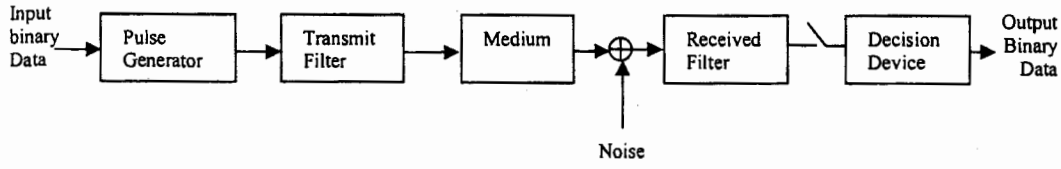


Fig 1.2 Base Band Model binary PAM

ISI can be reduced if the function $p(n)$ can be properly controlled. If the channel characteristics are known it is always possible to design a pair of transmitter and receiver that will minimize the effects of ISI. Using a fix pair of transmitters and receivers on the basis of average channel characteristics not necessary always reduce ISI. Adaptive Equalizers are used to control the effect of time invariant channels.

Equalization of data transmission system has different techniques and pre equalization at the transmitter and pre equalization at the receiver are one of them. The pre equalization at the transmitter needs feed back paths, so the adaptive filters are used at the receiver end to cancel the effect of ISI. The tap weights are adjusted in such a way the ISI is minimized and if these adjustable coefficients are made infinite the effect of ISI can be made very small. An adaptive algorithm needs the knowledge of desired signal or in theory the transmitted signal and is used to shape the error signal. The error signal is used for adaptive process to function.

The use of training sequence is one method to generate the desire response. A copy of the desired signal is stored in the receiver and synchronization is carried between this and the known transmitted sequence. The adaptive filtering algorithm with the help of the training sequence adjusts the equalizer coefficients and search for the unique minimum of error-performance surface. Second method is decision directed method in which a copy of the transmitted sequence is being produced at the output of the decision device in the receiver. If this output is the correct transmitted sequence, it may be used as the desired response for the purpose of Adaptive Equalization.

1.2. Blind Equalization

In blind equalization training sequence is not used. Non linear cost function is achieved at the output of the equalizer using the signal's structure and statistics of the data transmitted. The advantage of using blind equalizers is that there is no waste of initial transmitted data, longer time to converge is the drawback. Linear time invariant system is shown in Fig1.3, the input is x(n) and the symbols are independently and identically distributed and the probability distribution is known. The problem is to find x(n), and the channel given the output u(n).

Wireless communication systems need blind equalization because it is impractical to use the training sequence of long duration for two reasons. First one is that the cost to train the equalizer at the receiving end is high and second is the multipath fading is unavoidable. Blind equalization is used in a multipoint data network where data throughput needs to be increased and monitoring the network is to be made easy cause of the heavily loaded multipoint network. Another practical use of these algorithms is in reflection seismology, linear-predictive deconvolution is used to remove the source waveform from a seismogram.

There are two approaches to blind deconvolution, linear and nonlinear. Linear filters over samples the received signal changing received signal into multichannel signal. To find the input or desired response second-order statistics of the input signal are considered. Non linear filters are related to higher order statistics of the received

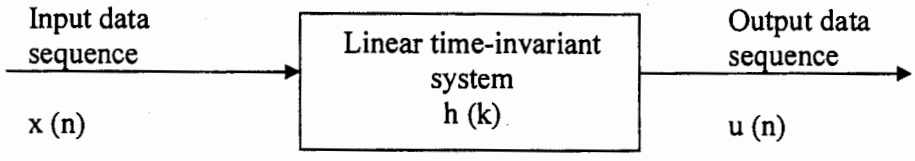


Fig. 1.3. Blind Deconvolution model

signal by the use of Bussgang algorithms. Blind equalizers are discussed by Godard [1] and Sato [2]. Bussgang methods are one of the suggested algorithms for the solution of blind equalization problems.

1.2.1 Sato Algorithm

Sato derived algorithm for multilevel pulse amplitude modulation (PAM), and Godard extended Sato's work for quadrature amplitude modulation (QAM) signals. Sato's proposed algorithm is given as follows [2], where E[.] represents exception, z(k) is transmitted data, $\hat{z}(k)$ is estimated signal and f(n) is output of transversal filter.

$$J(k) = E\,[\,(\,\hat{z}\,(k) - f(k)\,)^2\,] \tag{1.2}$$

The estimate $z\,(k)$ and constant γ is defined as,

$$z\,(k) = \gamma\ \mathrm{sgn}\,(\,f(k)\,) \tag{1.3}$$

$$\gamma = \frac{E\,[\,z(k)^2\,]}{E\,[\,abs(\,z(k)\,)\,]} \tag{1.4}$$

1.2.2. Constant Modulus Algorithm (CMA)

Godard derived an algorithm for blind equalization of QAM signals [2]. The cost function, update equation and the constant R_2 is defined as follows, where z(k) is the equalizer output, input data symbols are $\{a_k\}$ and μ is the set size parameter.

$$J = E\,[\,(\,abs(\,z(k)\,)^2 - R_2\,)^2\,] \tag{1.5}$$

$$C(k+1) = C(k) - \mu\, y(k)\, z(k)\,[\,(abs(\,z(k)\,)^2 - R_2\,] \tag{1.6}$$

$$R_2 = \frac{E\,[\,a(k)^4\,]}{E\,[\,abs(\,a(k)\,)^2\,]} \tag{1.7}$$

Chapter 2

Comparison of Bussgang Algorithms

2.1. Introduction

To overcome the problem of fading in wireless communication adaptive equalizers are used. The non effectiveness of using training sequence is well known [1], as a result blind equalization algorithms are employed. The algorithms introduced by Godard [1] and then the generalization of Sato's work [3] for complex data symbols have been applied and referred for QAM systems. Improved algorithms have also discussed in the literature [4-8]. White [9] also gave his approach for constant modulus algorithm (CMA). A model for blind equalization is given in Fig 2.1.

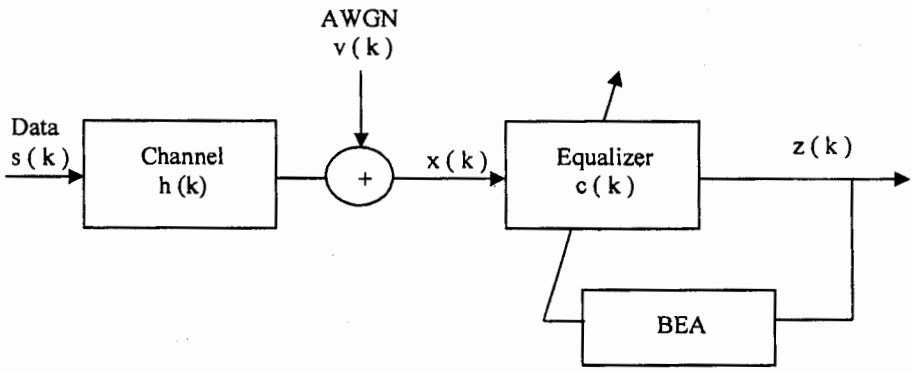


Fig. 2.1 Model - Blind Channel Equalization

2.2. Modified Constant Modulus Algorithm (MCMA)

The modified CMA cost function was proposed by Oh and Chin [10]. Its comparison with others is shown with the help of computer simulations in next chapter. The cost function is given as

$$J(n) = J_{Real}(n) + J_{Img}(n) \tag{2.1}$$

where $J_{Real}(n)$ *and* $J_{Img}(n)$ are real and imaginary parts of the cost function respectively. Equalizer output $z(n) = z_{Real}(n) + z_{Img}(n)$ are given as

$$J_{Real}(n) = E[(|z_{Real}(n)|^P - R_{p,Real})^2] \tag{2.2}$$

$$J_{Img}(n) = E[(|z_{Img}(n)|^P - R_{p,Img})^2] \tag{2.3}$$

$R_{p,Real}$ *and* $R_{p,Img}$ are real and imaginary constants respectively and input data sequence is s(k) is i.i.d. random variable.

$$R_{p,Real} = \frac{E[|s_{Real}(n)|^{2P}]}{E[|s_{Real}(n)|^P]} \tag{2.4}$$

$$R_{p,Img} = \frac{E[|s_{Img}(n)|^{2P}]}{E[|s_{Img}(n)|^P]} \tag{2.5}$$

Update equation is defined as

$$C(n+1) = C(n) - \mu.\hat{\nabla} J(n)$$

$$= C(n) - \mu . e(n) . X^*(n) \tag{2.6}$$

μ is the step size parameter and the error signal $e(n) = e_{Real}(n) + e_{Img}(n)$ is

$$e_{Real}(n) = z_{Real}(n) \, |z_{Real}(n)|^{P-2} \, (|z_{Real}(n)|^P - R_{p,Real}) \tag{2.7}$$

$$e_{Img}(n) = z_{Img}(n) \, |z_{Img}(n)|^{P-2} \, (|z_{Img}(n)|^P - R_{p,Img}) \tag{2.8}$$

2.3. New Multimodulus Blind Equalization Algorithm with Relaxation

Shafat Abrar and Ismail Shah suggested blind equalization technique for quadrature amplitude modulation (QAM) signals and the channel is assumed time-invariant [11], the next chapter contains computer simulations for comparison of their work with modified CMA [10].

Channel: $w_k = \sum_{n=0}^{N-1} h_n s_{k-n} + v_k$

Equalizer outputs: $z_k = c_k^T w_k = z_R(k) + j z_I(k)$

The channel input is s_k and additive noise is v_k. Using the method of Lagrange multipliers, cost function is

$$J = \min_{c_k} \| c_{k+1} - c_k \|_2^2 + \lambda_1 (| a_{R,k} |^p - R_R^p) + \lambda_2 (| a_{I,k} |^p - R_I^p) \qquad (2.9)$$

For p = 2 the above cost function becomes same as employed in [12] and [13]. Where c_{k+1} and a_k is given as

$$c_{k+1} = c_k + \mu (\varphi [z_k] - z_k) w_k^* \| w_k \|_2^{-2},$$

$$a_k = c_{k+1}^T w_k = \mu \varphi [z_k] + (1 - \mu) [z_k].$$

a_k comprises of the a priori output z_k and the blind estimate $\varphi [z_k]$ .The approach is to develop an algorithm by forcing posteriori output a_k and priori output z_k to come closer to the blind estimate $\varphi [z_k]$.

$$J = \min_{c_{k+1}} \{ \| c_{k+1} - c_k \|_2^2 + \lambda_1 (|a_{R,k}|^\Gamma |z_{R,k}|^q - R_R^{\Gamma+q}) + \lambda_2 (|a_{I,k}|^\Gamma |z_{I,n}|^q - R_I^{\Gamma+q}) \} \qquad (2.10)$$

where $\Gamma \geq 1$, $q \geq 1$. For derivation, we use $\Gamma = 2$ and $q = 2p - 2$.

We differentiate Eq. 2.10 with respect to c_{k+1}^* and set that to zero.

$$\partial J \; / \; \partial \, c_{k+1}^* \; = \; c_{k+1} \; - \; c_k + \lambda_1 \; | \; z_{R,k} \; |^{2p-2} \; a_{R,k} \; w_n^* \; + \; \lambda_2 \; | \; z_{I,k} \; |^{2p-2} \; a_{I,k} \; w_k^* \; = \; 0$$

Solving this gives the following update equation

$$c_{k+1} = c_k + \mu \left[\frac{z_{R,k} \; (1 - \; | \; z_{R,k} \; |^p \; / \; R_R^p)}{1 - \mu \; (1 - \; | \; z_{R,k} \; |^p \; / \; R_R^p)^{-1}} \; + j \; \frac{z_{I,k} \; (1 \; - \; | \; z_{I,k} \; |^p \; / \; R_I^p)}{1 - \mu \; (1 - | \; z_{I,k} \; |^p \; / \; R_I^p)^{-1}} \right] \frac{w_k^*}{\| \; w_k \; \|_2^2} \qquad (2.11)$$

If $\mu \; \square \; 1$, then we can assume

$$\mu \; (\; 1 - \; | \; z_{L,k} \; |^p \; / \; R_L^p) \approx \; 0$$

where L = R or I, to yield a much simpler expression given by

$$c_{k+1} \; = \; c_k \; + \; \mu \left[z_{R,k} \; (1 \; - \; | \; z_{R,k} \; |^p \; / \; R_R^p + j \; z_{R,k} \; (1 \; - \; | \; z_{I,k} \; |^p \; / \; R_I^p \right] \; \frac{w_k^*}{\| \; w_k \; \|_2^2}$$

$\| \; w_k \; \|_2^2$ do not help in convergence [14], based on this result it is removed. The four quadrant symmetry of QAM constellation allows us to write $R_R^p = R_I^p$, leading to

$$c_{k+1} = \; c_k \; + \; \mu \left[z_{R,k} (\; R_R^p \; - \; | \; z_{R,k} \; |^p) + j \; z_{I,k} (1 \; - \; | \; z_{I,k} \; |^p \; / \; R_I^p \right] \; w_k^* \qquad (2.12)$$

Eq. 2.12 is a new MMA that becomes the same as Eq. 2.6 for p = 2 where

$$R_R^p = \frac{E[| \; s_R \; |^{p+2}]}{E[| \; s_R \; |^p]} \text{ and}$$

$$R_I^p = \frac{E[| \; s_I \; |^{p+2}]}{E[| \; s_I \; |^p]}$$

2.4. Fractionally Spaced Bussgang Equalizers

The extension of New Multimodulus Blind Equalization Algorithm with Relaxation [11] is implemented by received antenna array or over sampling. The fractional space method has perfect equalizations and applied by received antenna array also it is used to directly estimate the equalizer W.

A model for received antenna array system is shown in Fig. 2.2, the proposed system uses four received antennas to perform simulations. A comparison between the above Modified Constant Modulus Techniques and the algorithm New Multimodulus Blind Equalization Algorithm with Relaxation using the Fractional Space Method for CMA is discussed in the next chapter in detail, with the help of simulations is shown that better performance of fractional space equalizers is achieved.

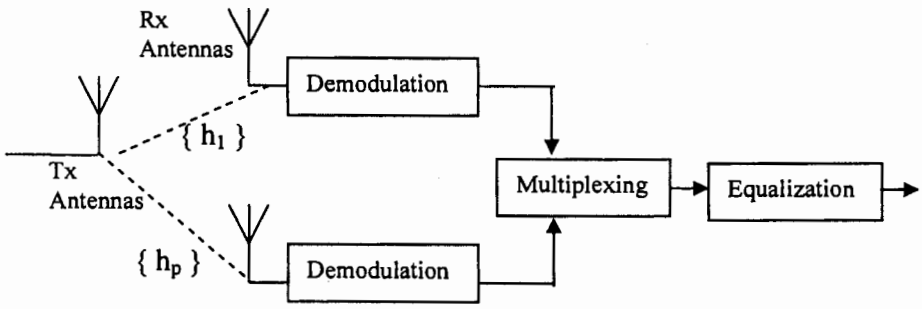


Fig. 2.2 Model of a received antenna array system

In the vector model shown in Fig. 2.3. s_n is the transmitted signal hi $_x$ is the

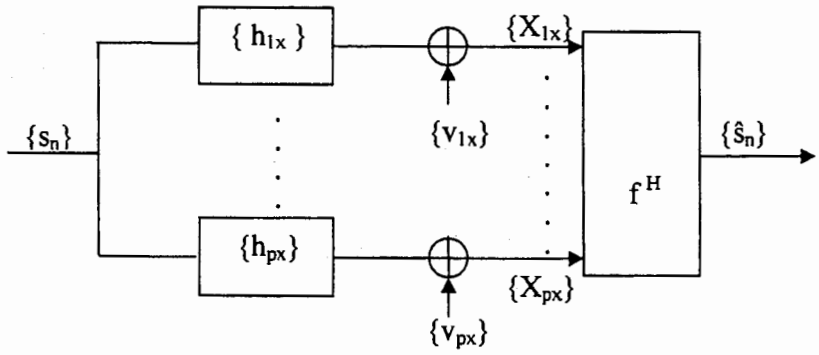


Fig. 2.3 Vector Model of a received antenna array system

channel impulse response where i =1, 2, ..., p for our case p = 4. $v_{i\,x}$ is the additive white guassian noise where i =1, 2, ... , p. $X_{i\,x}$ is the received signal at each antenna here i =1, 2, ..., p and f is the update vector and $\hat{s}_n$ is estimated signal. The matrix model shows the intermediate output after the signal is passed through the received antenna system where $h_{i\,j}$ is channel response where i =1, 2, ..., p and j =1, 2, ..., L.

2.4.1. Proposed Matrix Model

The basic matix model is given as follows,

$$\begin{bmatrix} X_{1n} \\ \vdots \\ X_{pn} \end{bmatrix} = \left[\begin{pmatrix} h_{11} & \cdots & h_{1L} \\ \vdots & \ddots & \vdots \\ h_{p1} & \cdots & h_{pl} \end{pmatrix} \right] \begin{bmatrix} s_{1n} \\ \vdots \\ s_{n-L} \end{bmatrix} + \begin{bmatrix} v_{1n} \\ \vdots \\ v_{pn} \end{bmatrix} \tag{2.13}$$

Expanded form of matrix model N+L input signals can be written as follows

$$\begin{bmatrix} X_{1n} \\ \vdots \\ X_{pn} \\ \vdots \\ \vdots \\ X_{1,n-N} \\ \vdots \\ X_{p,n-N} \end{bmatrix} = \begin{bmatrix} h_{11} & \cdots & h_{1L} & & & \\ \vdots & \ddots & \vdots & & & \\ h_{p1} & \cdots & h_{pl} & & & \\ & \ddots & & \ddots & & \\ & & \ddots & & \ddots & \\ & & & h_{11} & \cdots & h_{1L} \\ & & & \vdots & \ddots & \vdots \\ & & & h_{p1} & \cdots & h_{pl} \end{bmatrix} \begin{bmatrix} s_n \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ s_{n-N-L} \end{bmatrix} + \begin{bmatrix} v_{1n} \\ \vdots \\ v_{pn} \\ \vdots \\ \vdots \\ v_{1,n-N} \\ \vdots \\ v_{p,n-N} \end{bmatrix} \tag{2.14}$$

$$\underline{X}(n)_{[P\,(N+1)]\,x\,1} = \underline{H}_{[P\,(N+1)]\,x\,N+L+1}\ \underline{s}(n)_{[N+L+1]\,x\,1} + \underline{v}(n)_{[P\,(N+1)]\,x\,1} \tag{2.15}$$

Fractional space is similar to Bussgang Algorithms. In Fractional space it has Global convergence

$$\min J = E\,[(\,|\,f^{H}\,X(n)\,|\,^{2} - R_2\,)^2]$$

where update Equation is given by

$$f_{n+1} = f_n - \mu\,2\,[\,|\,f^{H}\,X(n)\,|\,^{2} - R_2\,]\,X(n)\,X^{H}\,(n)\,f_n \tag{2.16}$$

The constant R_2 is defined as $R_2 = \dfrac{E\,[\,s(n)^4\,]}{E\,[\,abs(\,s(n)\,)^2\,]}$ and f is the equalized output.

Chapter 3

Simulation and Results

3.1. Channels

Computer simulations have been used to compare the performance of the three Bussgang algorithms described in section 2.2, 2.3 and 2.4. In all simulations, complex taps transversal equalizers are used and are initialized so that the center tap was set to one. The channels used in the simulation are shown in Fig 3.1 and Fig 3.2 and assumed to be time invariant. Fig. 3.1.a shows the real part of channel I.

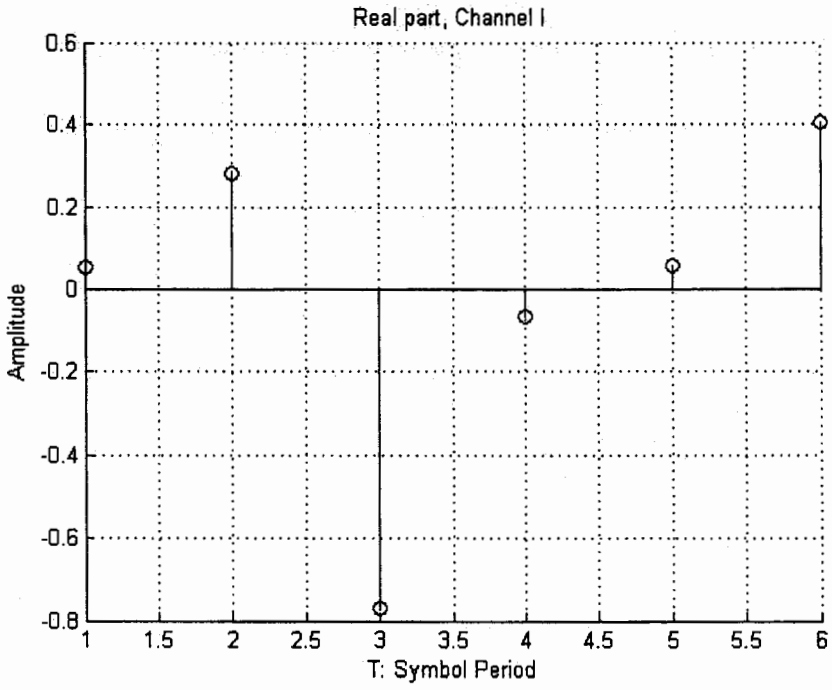


Fig. 3.1.a Real part of channel-I impulse response

Fig. 3.1.b shows the imaginary part of the complex channel I. The x-axis contains time period and y-axis shows the amplitude.

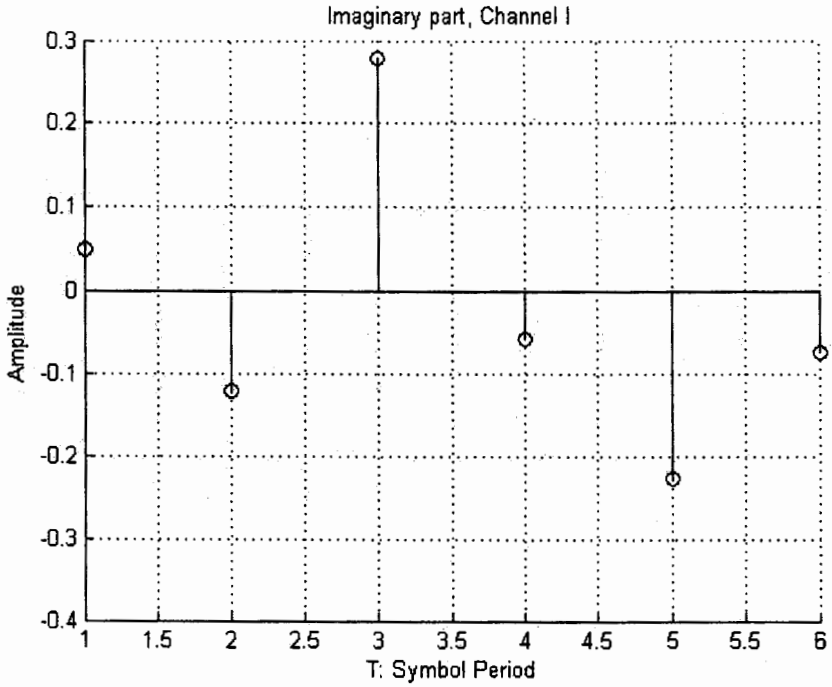


Fig. 3.1.b Imaginary part of channel-I impulse response

The channel shown in fig 3.2 is taken form [10]. The signal to noise ratio (SNR) was taken as 25dB at the input of the equalizer. Transmitted symbols are 4-QAM.

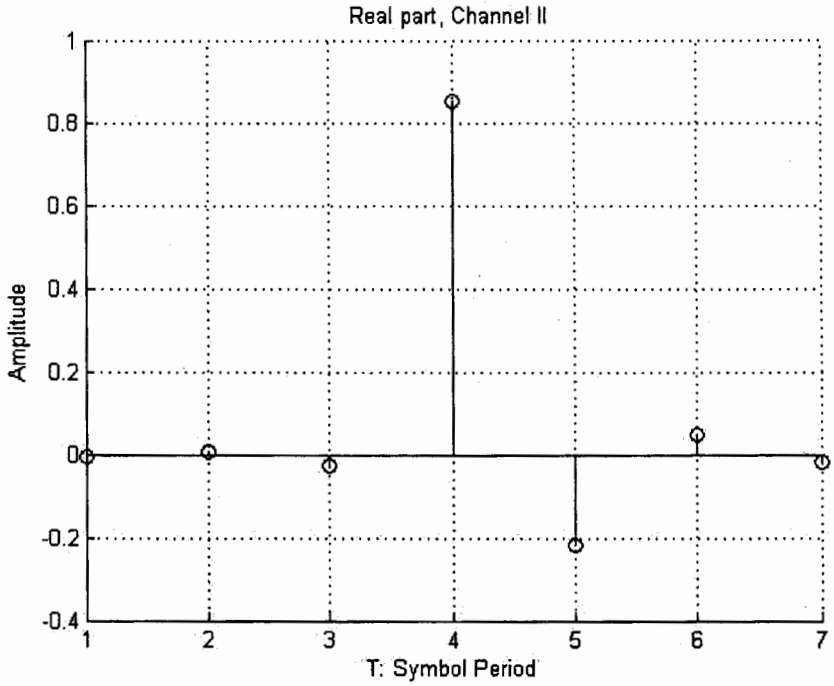


Fig. 3.2.a Real part of channel-II impulse response

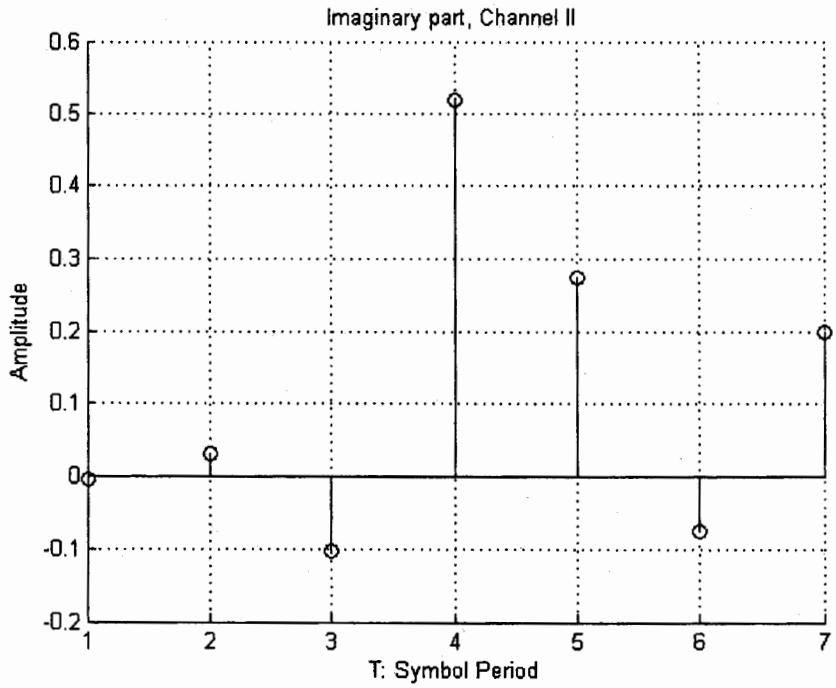


Fig. 3.2.b Imaginary part of channel-II impulse response

3.2. Received Equalized Outputs

Shown in Fig. 3.3.a, 3.3.b and 3.3.c are the output signals obtained after an equalizer with the modified CMA (M-CMA) previously proposed [10], plus New Multimodulus Blind Equalization Algorithm with Relaxation (NMBEAR) [11], and the proposed Fractional Space Method for CMA, respectively. The SNR is kept as 25dB the output in Fig. 3.3.c suggests the received signal implemented by received antenna array are more packed together and have better error performance e(n) for p = 5 in Eq. 2.6, 2.12 and 2.16 using the channel impulse response shown in Fig. 3.1.

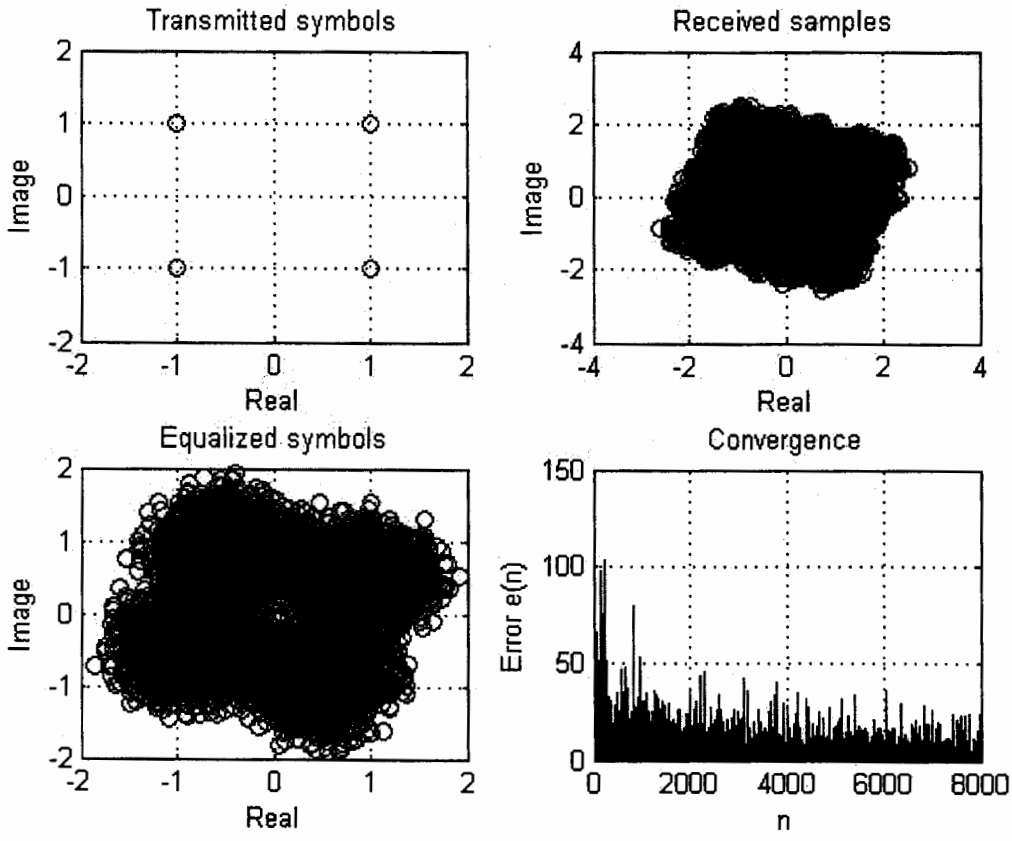


Fig 3.3.a. Received signal and error plot for Eq. 2.6 using Channel-I

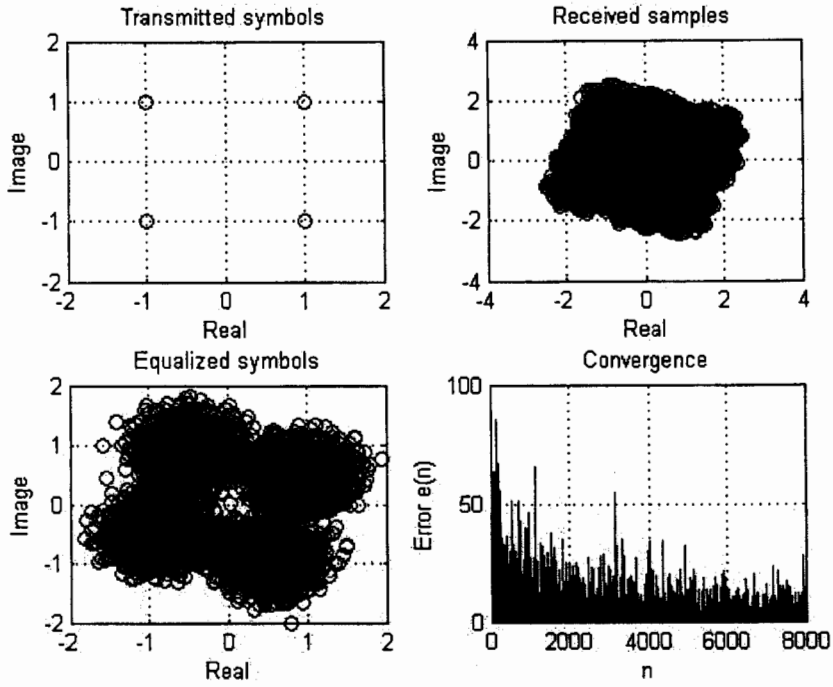


Fig 3.3.b. Received signal and error plot for Eq. 2.12 using Channel-I

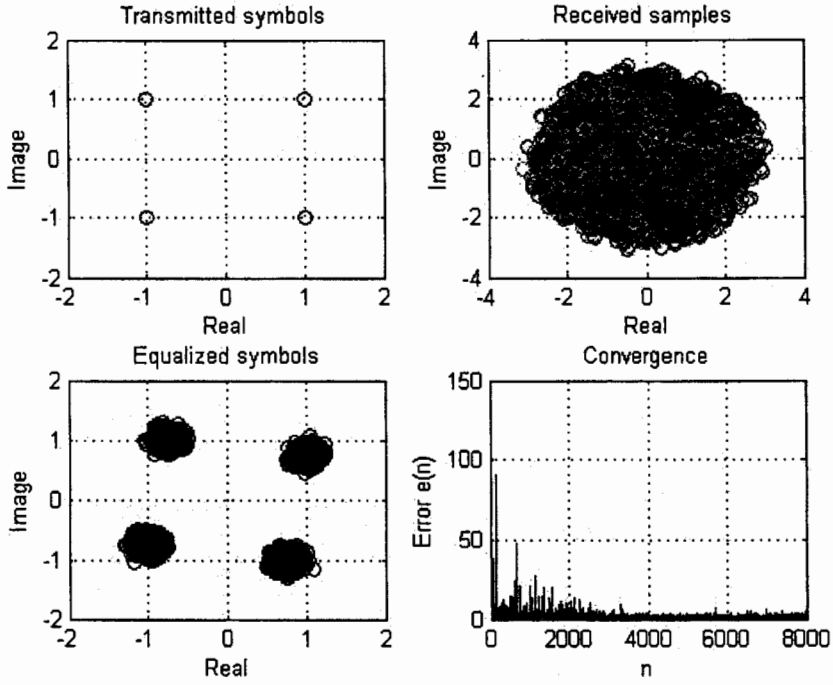


Fig 3.3.c. Received signal and error plot for Eq. 2.16 using received antenna array

Using channel impulse response shown in Fig. 3.2, comparison is done between Fig. 3.3.c for Fractional Space Method, Fig. 3.4.a for M-CMA and 3.4.b for NMBEAR, proposed algorithm shown in Fig. 3.3.c. gives better results. SNR is kept as 25dB and error performance e(n) is calculated for p = 5 for Eq. 2.6, 2.12 and 2.16.

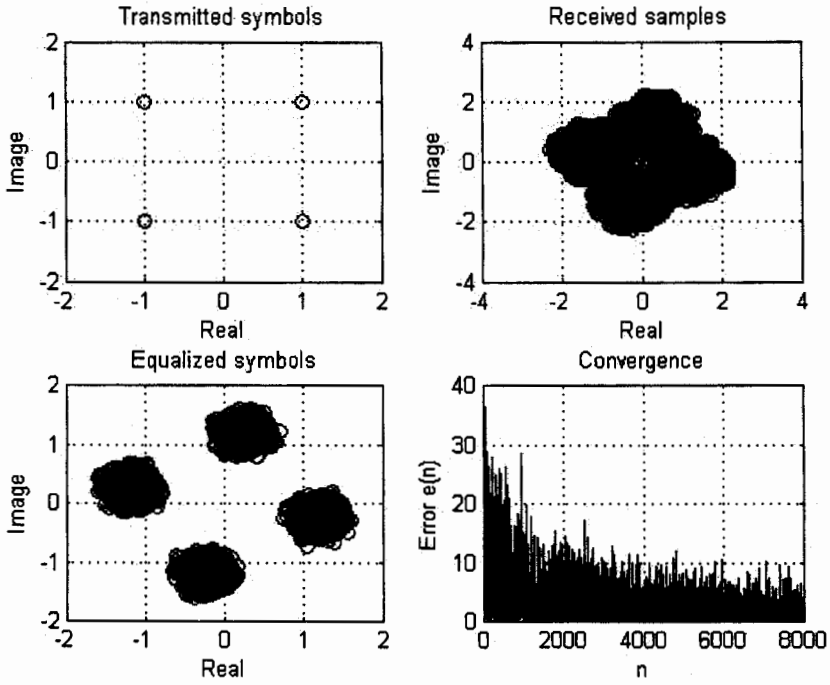


Fig 3.4.a. Received signal and error plot for Eq. 2.6 using Channel-II

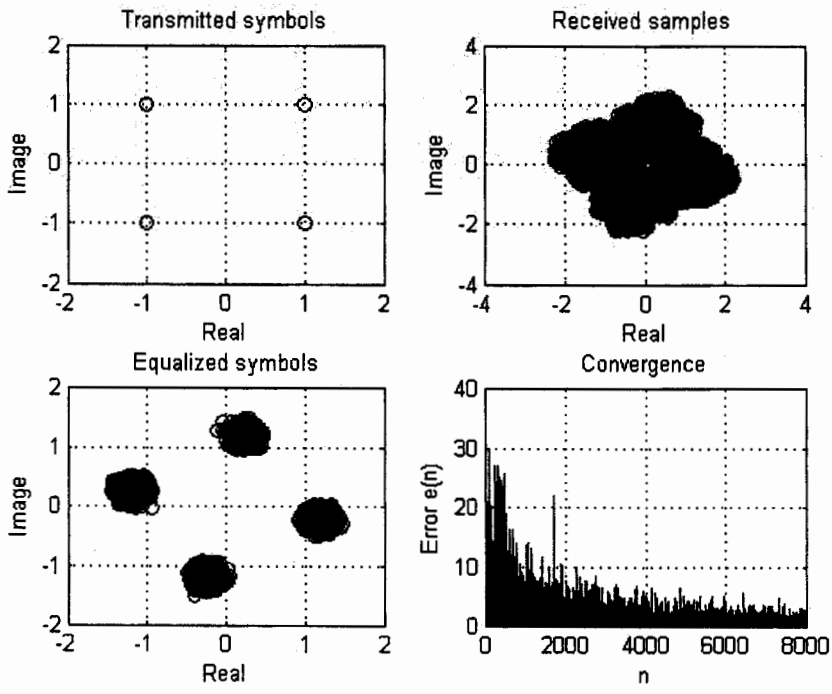


Fig 3.4.b. Received signal and error plot for Eq. 2.12 using Channel-II

3.3. Symbol Error Rate Plots

The signal-to-noise ratio (SNR) at the input of the equalizer is defined as

$$SNR = 10\log_{10}\frac{E\left[|a(k)*h(k)|^2\right]}{\sigma_n^2}$$

where σ_n^2 is the variance of the additive white noise, and ' * ' denotes convolution operation. The SER obtained over 100 of Monte Carlo trials by varying SNR from 5dB to 40 dB are shown in Fig. 3.5.a and 3.5.b for channel impulse response shown in Fig 3.1 and 3.2 respectively. The purposed algorithm given in Eq. 2.16 labeled in the figures as Multi Channel has better SER.

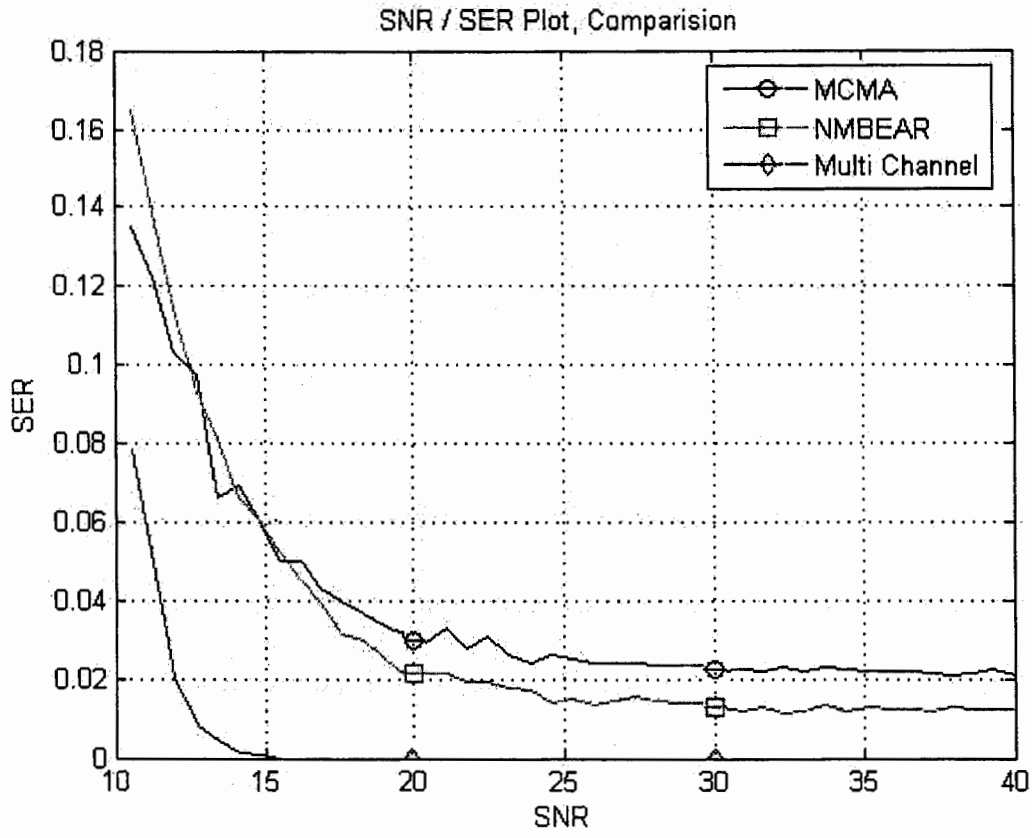


Fig. 3.5.a SER verses SNR graph comparison using Channel-I

Using for channel impulse response shown in Fig. 3.2 and by varying SNR from 5dB to 40 dB, the SER obtained over 100 of Monte Carlo trials. Fig 3.5.b. shows that the proposed signal converges quickly and has better performance. The SER goes to zero for the proposed system at SNR =11.

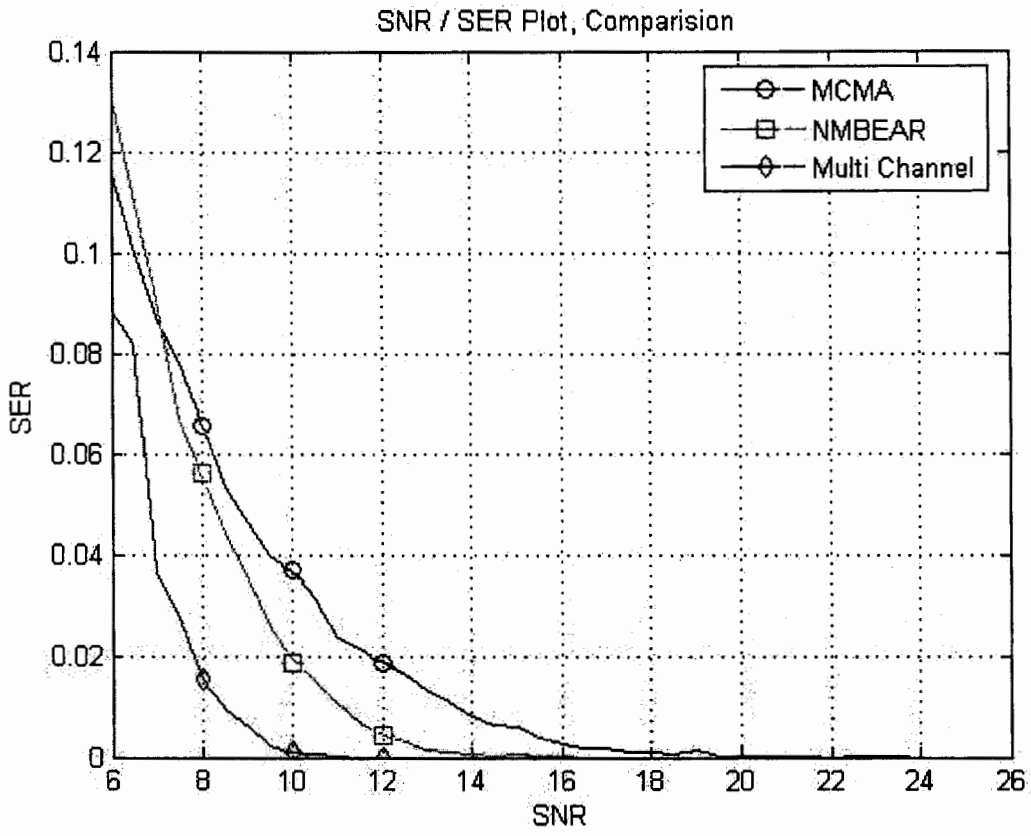


Fig. 3.5.b SER verses SNR graph comparison using Channel-II

The Epoc graphs 3.6.a and 3.6.b obtained over 200 independent trials, keeping the SNR constant 25 dB shows that the best results are obtained for among the compared are for over sampling given in Eq, 2.16.

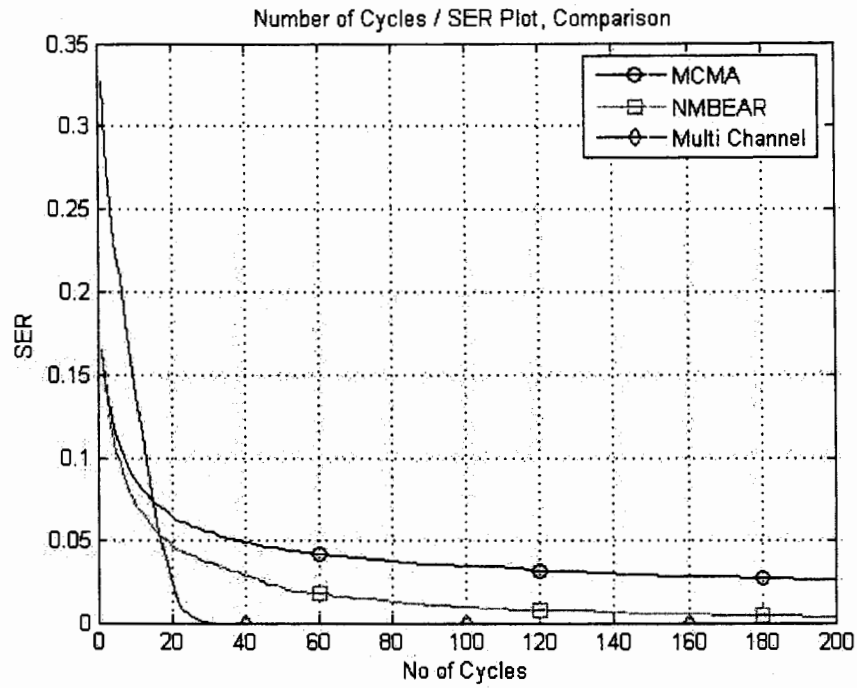


Fig 3.6.a SER calculated over 200 1independent trails for Channel-I

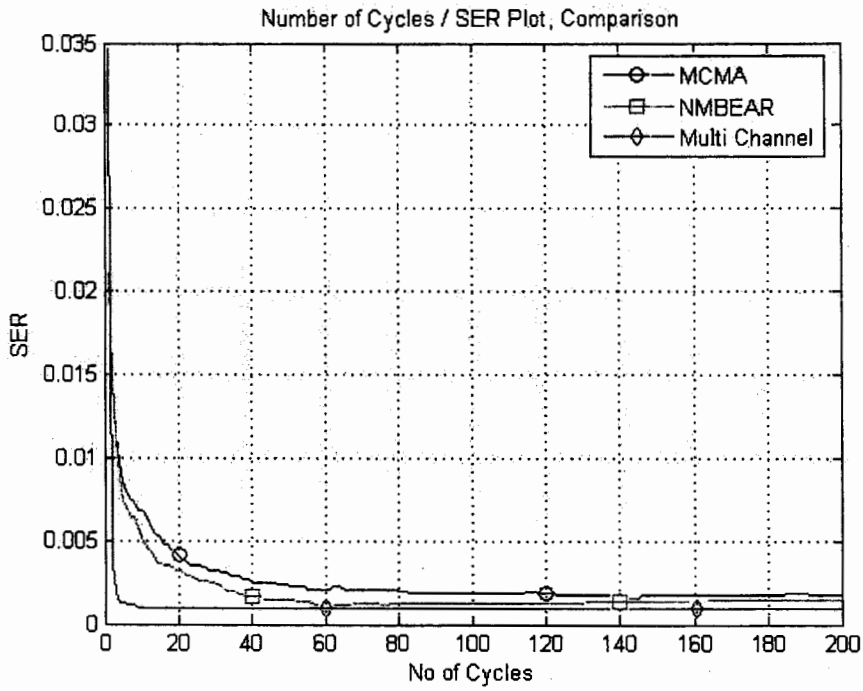


Fig 3.6.b SER calculated over 200 independent trails for Channel-II

3.4. Mean Square Error Plots

MSE is measures as follows

$$MSE(n) = \frac{1}{L}\sum_{l=1}^{L}[\, y(n) \; - \; a(n) \,)$$

where a(n) is the desired output of the equalizer at time index n. Similarly, y(n) is the equalizer output at time index n. The MSE obtained over 100 of Monte Carlo trials by varying SNR from 5dB to 40 dB are shown in Fig. 3.7.a and 3.7.b for channel impulse response shown in Fig 3.1 and 3.2 respectively

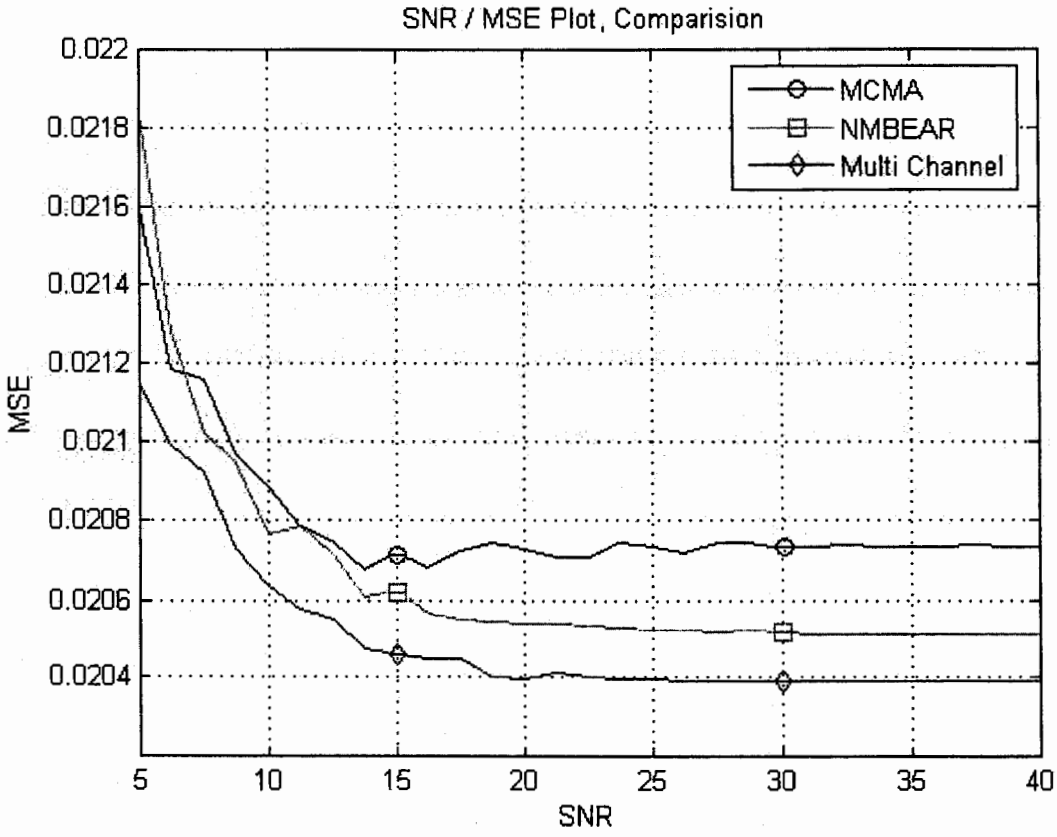


Fig. 3.7.a MSE verses SNR graph comparison using Channel-I

Using for channel impulse response shown in Fig. 3.2 and by varying SNR from 5dB to 40 dB, the MSE obtained over 100 of Monte Carlo trials. Fig 3.7.b. shows that the proposed signal converges quickly and has better performance. The MSE stabilizes to a constant value quickly for the proposed system.

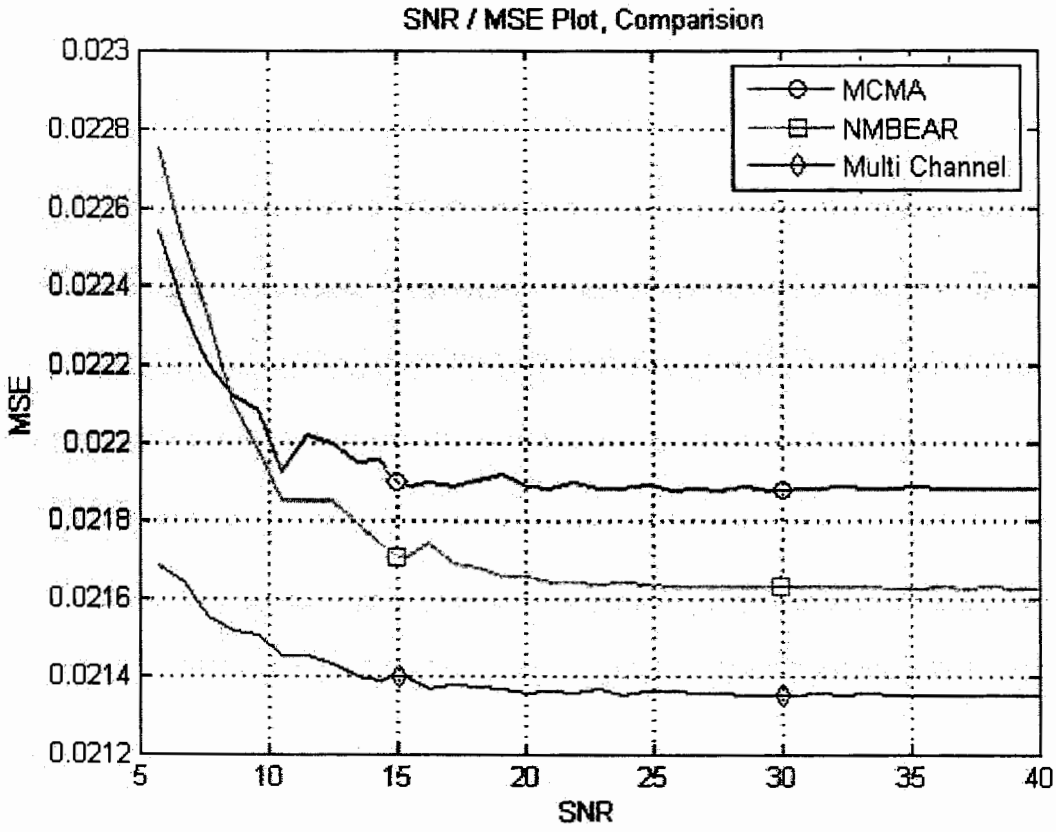


Fig. 3.7.b MSE verses SNR graph comparison using Channel-II

The results obtained are shown in Fig. 3.8.a and 3.8.b for ensemble averaged over 200 independent runs with an SNR of 25 dB. A 4-QAM data set has been used with a typical communication channel obtained from Fig 3.1 and Fig 3.2.

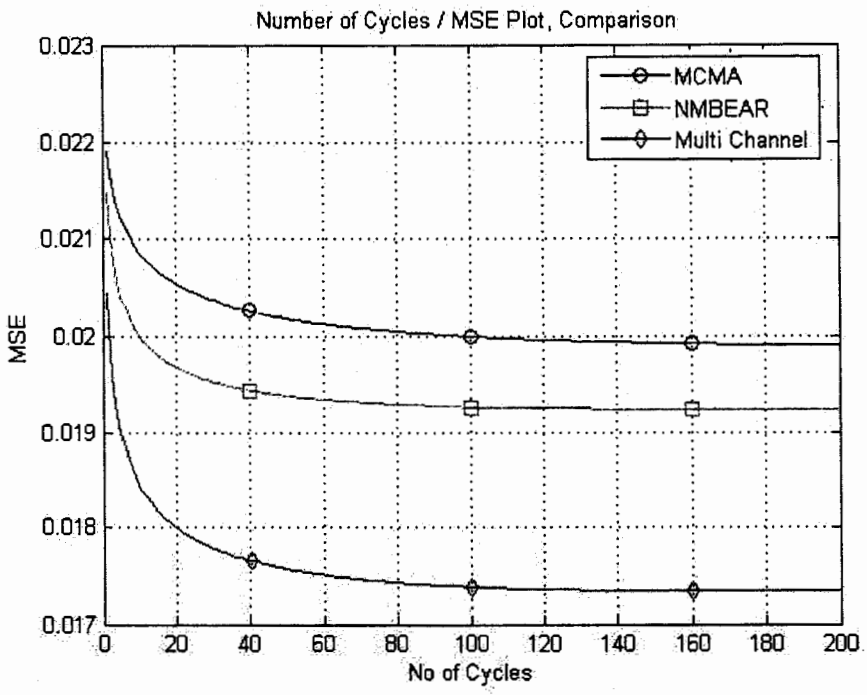


Fig 3.8.a MSE calculated over 200 independent trails for Channel-I

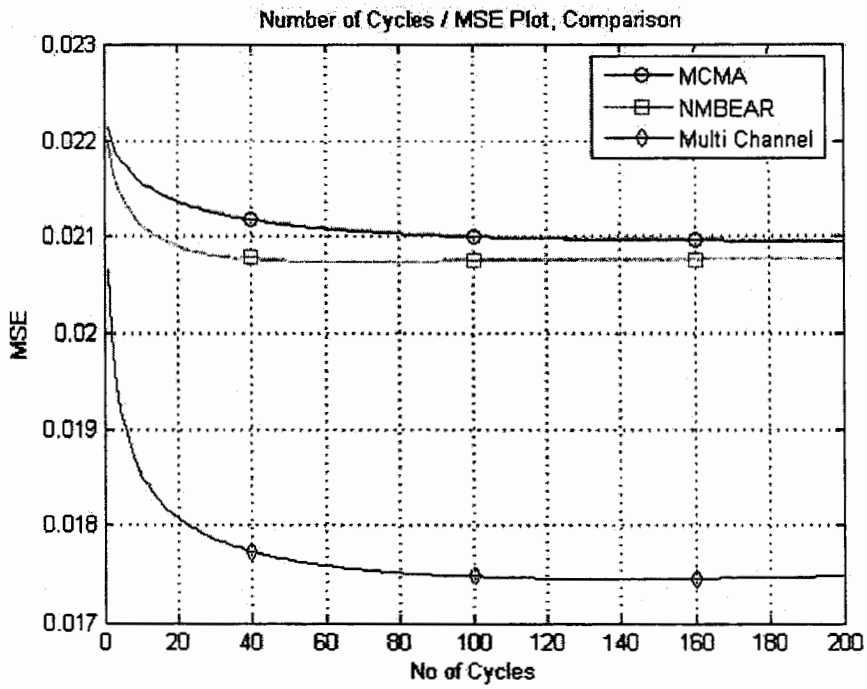


Fig 3.8.b MSE calculated over 200 independent trails for Channel -II

3.5. Normalized Root Mean Square ISI Plots

Next, we compare the convergence behavior and normalized root mean square IS1 of the algorithms discussed above. As the measure of performance, we use the NRMS IS1 at the output of the equalizer, which is defined by

$$NRMS - ISI = \frac{1}{\|s\|}\sqrt{\frac{1}{N_m}\sum_{i=1}^{N}\left\|s - \alpha_i \hat{s}\right\|^2}$$

N_m = No of Mote Carlo trials

N = Total no of symbols

s = Actual symbol

$\hat{s}$ = Estimated symbol

α = Rotational constant

$\|\,.\,\|$ = Indicates inner product

Fig. 3.9.a and 3.9.b shows the ensemble-averaged NRMS ISI, obtained from 100 Monte Carlo runs for 4-QAM constellation and SNR = 25 dB were chosen. From the results, we can see that the Modified CMA and N-MBEAR has slower convergence rate. Since the N-MBEAR implemented received antenna array removes IS1, it is possible to estimate the error more accurately. Thus, the new algorithm results in performance enhancement in convergence speed and NRMS IS1 than those of the M-CMA and N-MBEAR.

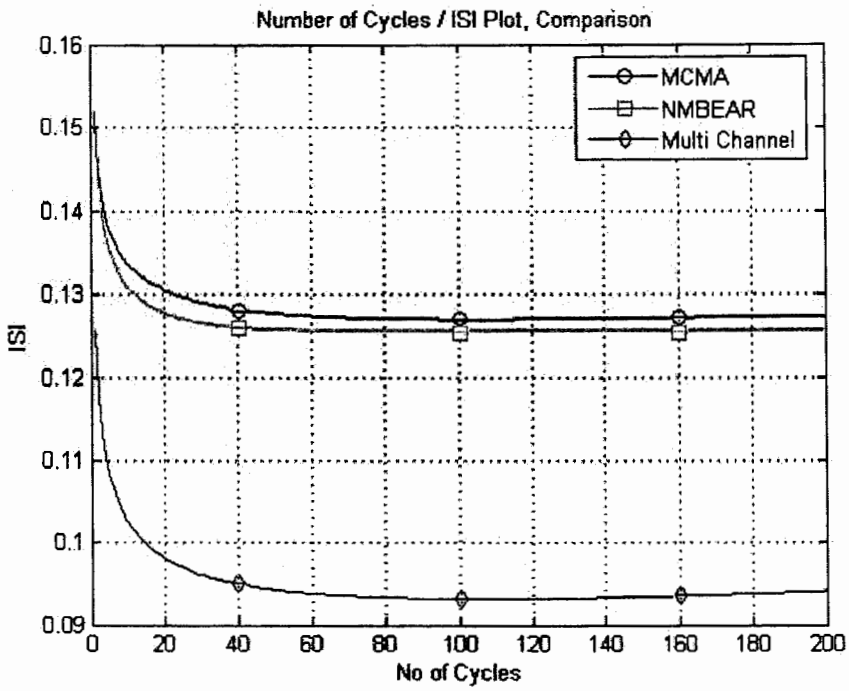


Fig 3.9.a ISI calculated over 200 independent trails for Channel-I

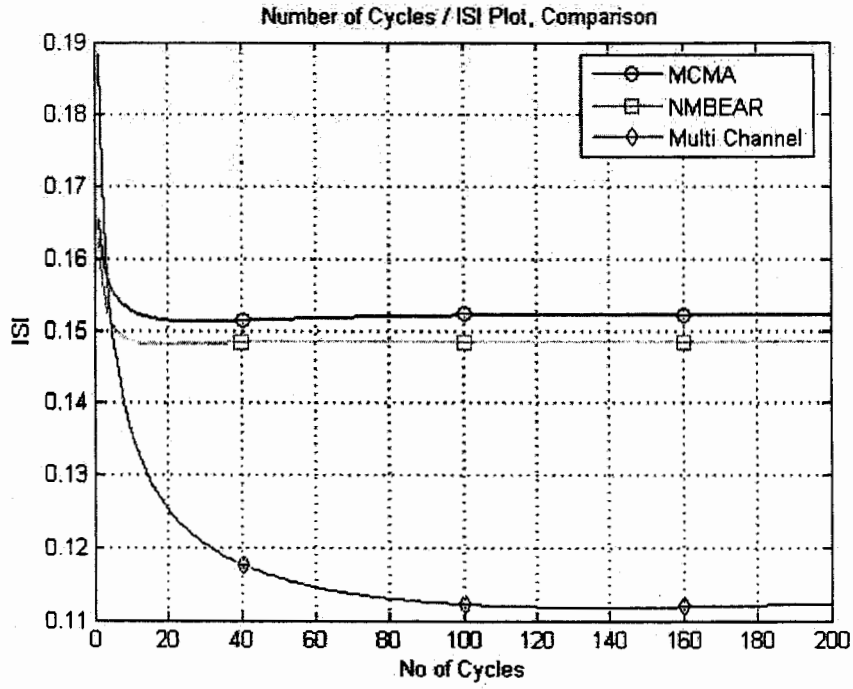


Fig 3.9.b ISI calculated over 200 independent trails for Channel-II

To corroborate the above discussion, we simulated a 4-QAM constellation in a channel shown in Fig. 3.1 and 3.2 respectively with additive white Gaussian noise SNR 25dB. The Normalized Root Mean Square ISI is measured by averaging 200 independent realizations of noise and data source. In Fig. 3.10.a and 3.10.b, the ISI traces obtained from Eq. 2.6 are depicted for p=1, 2, 3, 4 and 5. However, for p > 5, the Modified Constant Modulus Algorithm (M-CMA) algorithm given in Eq.2.6 yielded no stable convergence for any value of step-size.

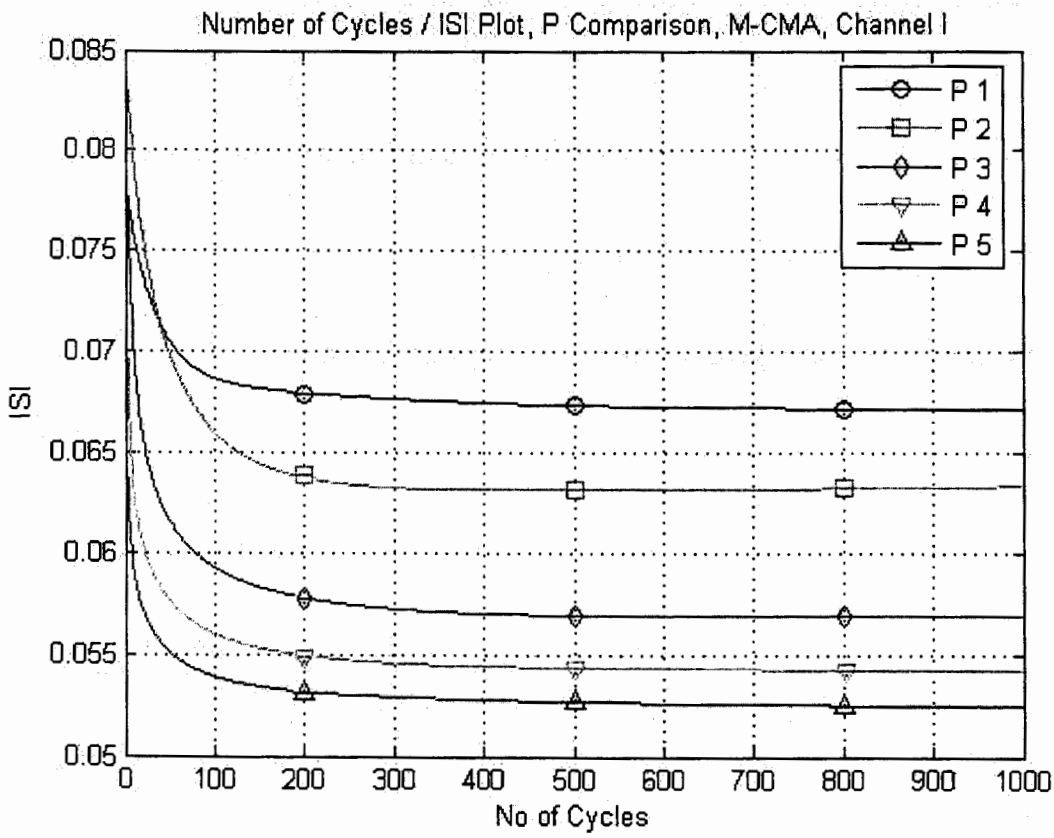


Fig 3.10.a. For values of Multimodulus (P), NRMS-ISI is calculated using M-CMA and Channel-I

Similarly when the channel II shown in Fig. 3.2 is used for p > 5, the Modified Constant Modulus Algorithm (M-CMA) algorithm described in Eq. 2.6 yielded no stable convergence for any value of step-size. Fig 3.10.b shows M-CMA implemented for P= 1, 2,…,5

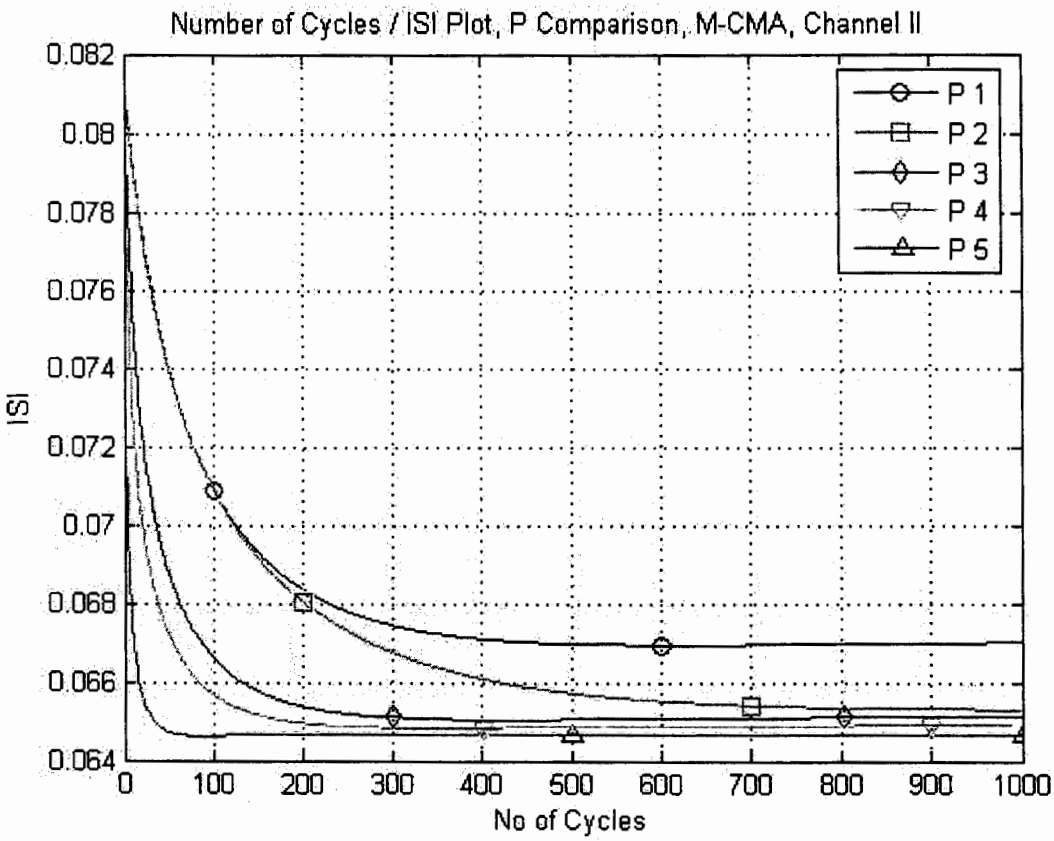


Fig 3.10.b. For values of Multimodulus (P), NRMS-ISI is calculated using M-CMA and Channel-II

In Fig. 3.10.c, the ISI traces obtained from (2.12) are depicted for p=1, 2, 3, 4, 5, 6 and 7. However, for p > 7, the algorithm given in Eq. 2.12 yielded no stable convergence for any value of step-size using channel shown in Fig. 3.1 and 3.2 respectively. On the other hand, as depicted in Fig. 3.11, the proposed algorithm derived in Eq. 2.16 is yielding stable convergence for p = 1 to 9 with consistent lowering in ISI floor (without affecting the convergence speed).

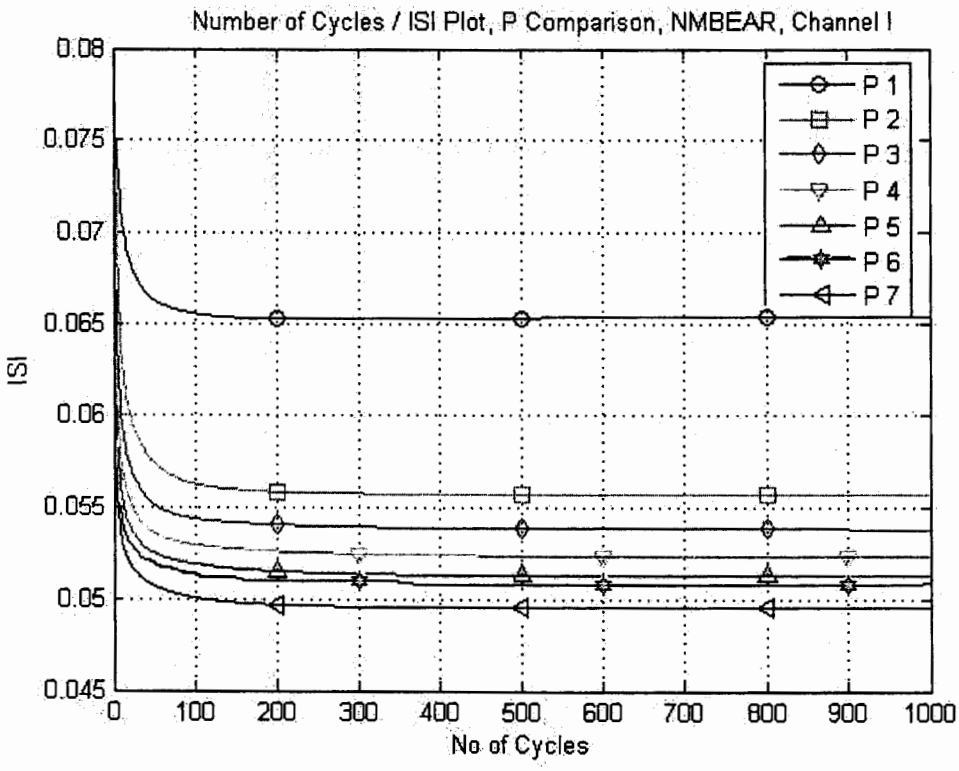


Fig 3.10.c. For values of Multimodulus (P), NRMS-ISI is calculated using NMBEAR and Channel-I

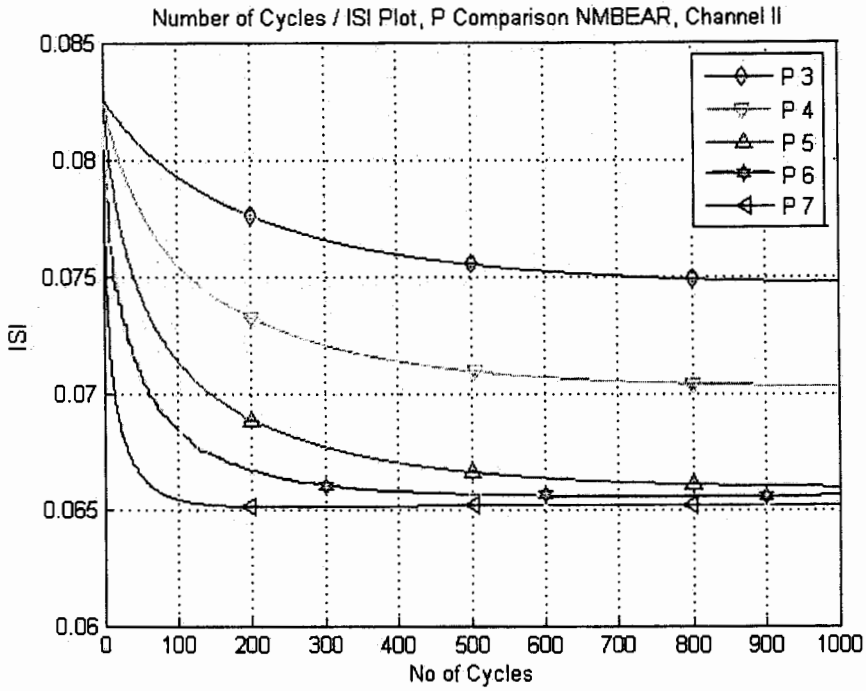


Fig 3.10.d. For values of Multimodulus (P), NRMS-ISI is calculated using NMBEAR and Channel-II

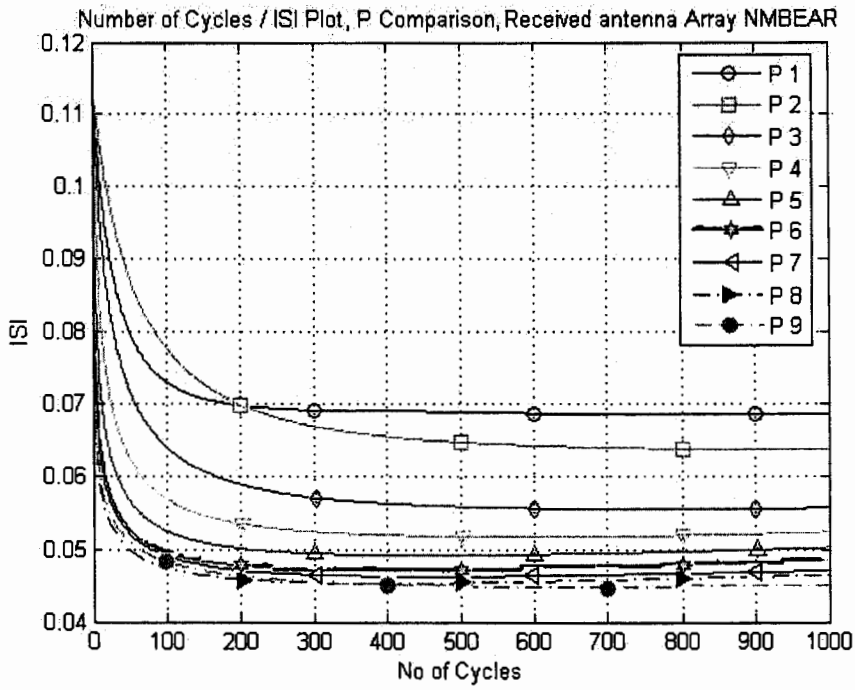


Fig 3.11 For values of Multimodulus (P), NRMS-ISI is calculated using Received antenna Array NMBEAR

TH-4452

Chapter 4

Conclusion

4.1. Simulation results

Blind equalizer compared to training based equalizer, is able to cope with amplitude and delay distortion of a communication channel using channel output sample and statistical properties of the data symbol. The three blind channel equalization algorithm Modified Constant Modulus algorithm, New Multimodulus blind equalization algorithm with relaxation (NMBEAR) and using Fractional Space Method for New Multimodulus blind equalization algorithm with relaxation have been implemented for 4-QAM signal constellation. The signals received by antenna array are more packed together and have better error performance. The curves of SER, MSE and ISI verses SNR and by averaging 200 independent realizations of noise and data source have shown that the Fractional Space method yields better performance than other algorithms.

In Fig. 3.10, the ISI traces obtained from MCMA and NMBERA are depicted for p=1 to 5 and p=1 to 7 respectively. However, for p > 5, the algorithms yielded no stable convergence for any value of step-size. On the other hand, as depicted in Fig. 3.11, the Received antenna Array NMBEAR algorithm is yielding stable convergence for p =1 to 9 with consistent lowering in ISI floor (without affecting the convergence speed).

4.2. Future Work

Channel equalization techniques are getting sophisticated with time and the current work to minimize the disadvantages to blind equalization needs further research. Algorithms are designed to overcome ISI and also have faster convergence, better steady state error and phase recovery.

In the previous chapters, the algorithm was designed for four multiple received channels. It is proposed that future work can be done by taking more multiple channels into account to yield stable convergence for $P > 9$ with consistent lowering the ISI floor and without affecting the convergence speed. The comparison can be extended to use the subspace method for blind equalization and rake receivers for CDMA to accomplish better analysis of interference cancellation. The resulting cost functions compared for the above algorithms can correct the phase error and removes ISI, to estimate the error accurately the automatic phase recovery can be compared.

Appendix

MATLAB Code

A.1. Simulation of the Received signal and error performance e(n)

```
T=8000;   % total number of data
dB=25;    % SNR in dB value
%error performance and plotting of received signal
%%%%%%%%%% Simulate the Received noisy Signal  %%%%%%%%%%%%
N=20; % smoothing length N+1
%Lh=5;  % channel length = Lh+1 for Channel-I
Lh=6;  % channel length = Lh+1
P=round((N+Lh)/2);  % equalization delay
% channel-II shown in figure 3.2 (complex)
j=sqrt(-1);
h=[-0.005-j*0.004 .009+.03*j -.024-.104*j .854+.520*j ...
     -.218+.273*j .049-.074*j -.016+.20*j];
%Channel- I shown in figure 3.1
%h=[0.0545+j*0.05 .2832-.1197*j -.7676+.2788*j -.0641-.0576*j ...
     %.0566-.2275*j .4063-.0739*j];
h=h/norm(h);                % normalize
s=round(rand(1,T))*2-1;  % QPSK or 4 QAM symbol sequence
s=s+sqrt(-1)*(round(rand(1,T))*2-1);
% generate received noisy signal
x=filter(h,1,s);
vn=randn(1,T)+sqrt(-1)*randn(1,T);  % AWGN noise (complex)
vn=vn/norm(vn)*10^(-dB/20)*norm(x);  % adjust noise power with SNR dB value
SNR=20*log10(norm(x)/norm(vn))      % Check SNR of the received samples
x=x+vn;                     % received signal
%%%%%%%%%%%%%% adaptive equalizer estimation via CMA
Lp=T-N;  %% remove several first samples to avoid 0 or negative subscript
X=zeros(N+1,Lp);  % sample vectors (each column is a sample vector)
for i=1:Lp
  X(:,i)=x(i+N:-1:i).';
end
e=zeros(1,Lp);  % used to save instant error
f=zeros(N+1,1); f(P)=1;   % initial condition
%for P = 1
mu= 0.001;      % parameter to adjust convergence and steady error
R2=1.9;             % constant modulas of QPSK symbols
for i=1:Lp
  %for P = 1
  e(i)=abs(f'*X(:,i))-R2;              % instant error
  f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
  f(P)=1;
  i_e=[i/10000 abs(e(i))] ;          % output information
end
sb=f'*X;  % estimate symbols (perform equalization)
% calculate SER
H=zeros(N+1,N+Lh+1);
for i=1:N+1,
  H(i,i:i+Lh)=h;
end % channel matrix

fh=f'*H ;  % composite channel+equalizer response should be delta-like
```

```
temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response
sb1=sb/(fh(temp));  % scale the output
sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
start=6 ;  % carefully find the corresponding begining point
sb2=sb1-s(start+1:start+length(sb1));  % find error symbols
SER=length(find(sb2~=0))/length(sb2)   % calculate SER
figure
if 1
   subplot(221),
   plot(s,'o');  % show the pattern of transmitted symbols
   grid,title('Transmitted symbols');  xlabel('Real'),ylabel('Image')
   axis([-2 2 -2 2])
   subplot(222),
   plot(x,'o');  % show the pattern of received samples
   grid, title('Received samples');  xlabel('Real'), ylabel('Image')
   subplot(223),
   plot(sb,'o');  % show the pattern of the equalized symbols
   grid, title('Equalized symbols'), xlabel('Real'), ylabel('Image')
   axis([-2 2 -2 2])
   subplot(224),
   plot(abs(e));  % show the convergence
   grid, title('Convergence'), xlabel('n'), ylabel('Error e(n)')
end
Z=zeros(6,7980);
Z(1,:) = abs(e);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e=zeros(1,Lp);  % used to save instant error
f=zeros(N+1,1); f(P)=1;   % initial condition
%for P = 2
mu=0.001;     % parameter to adjust convergence and steady error
R2=2.2;              % constant modulas of QPSK symbols

for i=1:Lp
   %for P = 2
   e(i)=abs(f'*X(:,i))^2-R2;              % instant error
   f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f ;   % update equalizer
   f(P)=1;
   i_e=[i/10000 abs(e(i))] ;          % output information
end

sb=f'*X;  % estimate symbols (perform equalization)
% calculate SER
H=zeros(N+1,N+Lh+1);
for i=1:N+1,
   H(i,i:i+Lh)=h;
end  % channel matrix

fh=f'*H ;  % composite channel+equalizer response should be delta-like
temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response
sb1=sb/(fh(temp));  % scale the output
sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
start=6 ;  % carefully find the corresponding begining point
sb2=sb1-s(start+1:start+length(sb1));  % find error symbols
SER=length(find(sb2~=0))/length(sb2)   % calculate SER

figure
if 1
   subplot(221),
   plot(s,'o');  % show the pattern of transmitted symbols
   grid,title('Transmitted symbols');  xlabel('Real'),ylabel('Image')
```

```
  axis([-2 2 -2 2])
  subplot(222),
  plot(x,'o'); % show the pattern of received samples
  grid, title('Received samples'); xlabel('Real'), ylabel('Image')
  subplot(223),
  plot(sb,'o'); % show the pattern of the equalized symbols
  grid, title('Equalized symbols'), xlabel('Real'), ylabel('Image')
  subplot(224),
  plot(abs(e)); % show the convergence
  grid, title('Convergence'), xlabel('n'), ylabel('Error e(n)')
end
Z(2,:) = abs(e);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e=zeros(1,Lp); % used to save instant error
f=zeros(N+1,1); f(P)=1; % initial condition
%for P = 3
mu=0.0001;    % parameter to adjust convergence and steady error
R2=2.4;           % constant modulas of QPSK symbols
for i=1:Lp
  %for P = 3
  e(i)=abs(f'*X(:,i))^3-R2;          % instant error
  f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;   % update equalizer
  f(P)=1;
  i_e=[i/10000 abs(e(i))] ;         % output information
end

sb=f'*X; % estimate symbols (perform equalization)
% calculate SER
H=zeros(N+1,N+Lh+1);
for i=1:N+1,
  H(i,i:i+Lh)=h;
end % channel matrix

fh=f'*H ; % composite channel+equalizer response should be delta-like
temp=find(abs(fh)==max(abs(fh))); % find the max of the composite response
sb1=sb/(fh(temp)); % scale the output
sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1)); % perform symbol detection
start=6 ; % carefully find the corresponding begining point
sb2=sb1-s(start+1:start+length(sb1)); % find error symbols
SER=length(find(sb2~=0))/length(sb2) % calculate SER

figure
if 1
  subplot(221),
  plot(s,'o'); % show the pattern of transmitted symbols
  grid,title('Transmitted symbols'); xlabel('Real'),ylabel('Image')
  axis([-2 2 -2 2])
  subplot(222),
  plot(x,'o'); % show the pattern of received samples
  grid, title('Received samples'); xlabel('Real'), ylabel('Image')
  subplot(223),
  plot(sb,'o'); % show the pattern of the equalized symbols
  grid, title('Equalized symbols'), xlabel('Real'), ylabel('Image')
  subplot(224),
  plot(abs(e)); % show the convergence
  grid, title('Convergence'), xlabel('n'), ylabel('Error e(n)')
end
Z(3,:) = abs(e);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e=zeros(1,Lp); % used to save instant error
```

```
f=zeros(N+1,1); f(P)=1;   % initial condition
%for P = 4
mu=0.00004;     % parameter to adjust convergence and steady error
R2=2.93;              % constant modulas of QPSK symbols
for i=1:Lp
  %for P = 4
  e(i)=abs(f'*X(:,i))^4-R2;              % instant error
  f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
  f(P)=1;
  i_e=[i/10000 abs(e(i))] ;          % output information
end

sb=f'*X;  % estimate symbols (perform equalization)
% calculate SER
H=zeros(N+1,N+Lh+1);
for i=1:N+1,
  H(i,i:i+Lh)=h;
end  % channel matrix
fh=f'*H ;  % composite channel+equalizer response should be delta-like
temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response
sb1=sb/(fh(temp));  % scale the output
sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
start=6 ;  % carefully find the corresponding begining point
sb2=sb1-s(start+1:start+length(sb1));  % find error symbols
SER=length(find(sb2~=0))/length(sb2)  % calculate SER
figure
if 1
  subplot(221),
  plot(s,'o');  % show the pattern of transmitted symbols
  grid,title('Transmitted symbols');  xlabel('Real'),ylabel('Image')
  axis([-2 2 -2 2])
  subplot(222),
  plot(x,'o');  % show the pattern of received samples
  grid, title('Received samples');  xlabel('Real'), ylabel('Image')
  subplot(223),
  plot(sb,'o');  % show the pattern of the equalized symbols
  grid, title('Equalized symbols'), xlabel('Real'), ylabel('Image')
  subplot(224),
  plot(abs(e));  % show the convergence
  grid, title('Convergence'), xlabel('n'), ylabel('Error e(n)')
end
Z(4,:) = abs(e);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e=zeros(1,Lp);  % used to save instant error
f=zeros(N+1,1); f(P)=1;   % initial condition
%for P = 5
mu=0.00001;     % parameter to adjust convergence and steady error
R2=3;              % constant modulas of QPSK symbols
for i=1:Lp
  %for P = 5
  e(i)=abs(f'*X(:,i))^5-R2;              % instant error
  f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
  f(P)=1;
  i_e=[i/10000 abs(e(i))] ;          % output information
end

sb=f'*X;  % estimate symbols (perform equalization)
% calculate SER
H=zeros(N+1,N+Lh+1);
for i=1:N+1,
```

```
  H(i,i:i+Lh)=h;
end % channel matrix
fh=f'*H ;  % composite channel+equalizer response should be delta-like
temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response
sb1=sb/(fh(temp));  % scale the output
sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
start=6 ;  % carefully find the corresponding begining point
sb2=sb1-s(start+1:start+length(sb1));  % find error symbols
SER=length(find(sb2~=0))/length(sb2)  % calculate SER
figure
if 1
  subplot(221),
  plot(s,'o');  % show the pattern of transmitted symbols
  grid,title('Transmitted symbols');  xlabel('Real'),ylabel('Image')
  axis([-2 2 -2 2])
  subplot(222),
  plot(x,'o');  % show the pattern of received samples
  grid, title('Received samples');  xlabel('Real'), ylabel('Image')
  subplot(223),
  plot(sb,'o');  % show the pattern of the equalized symbols
  grid, title('Equalized symbols'), xlabel('Real'), ylabel('Image')
  axis([-2 2 -2 2])
  subplot(224),
  plot(abs(e));  % show the convergence
  grid, title('Convergence'), xlabel('n'), ylabel('Error e(n)')
end
Z(5,:) = abs(e);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e=zeros(1,Lp);  % used to save instant error
f=zeros(N+1,1); f(P)=1;   % initial condition
%for P = 6
mu=0.000009;     % parameter to adjust convergence and steady error
R2=3.1;          % constant modulas of QPSK symbols
for i=1:Lp
  %for P = 6
  e(i)=abs(f'*X(:,i))^6-R2;              % instant error
  f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
  f(P)=1;
  i_e=[i/10000 abs(e(i))] ;          % output information
end
sb=f'*X;  % estimate symbols (perform equalization)
% calculate SER
H=zeros(N+1,N+Lh+1);
for i=1:N+1,
  H(i,i:i+Lh)=h;
end % channel matrix
fh=f'*H ;  % composite channel+equalizer response should be delta-like
temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response
sb1=sb/(fh(temp));  % scale the output
sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
start=6 ;  % carefully find the corresponding begining point
sb2=sb1-s(start+1:start+length(sb1));  % find error symbols
SER=length(find(sb2~=0))/length(sb2)  % calculate SER

figure
if 1
  subplot(221),
  plot(s,'o');  % show the pattern of transmitted symbols
  grid,title('Transmitted symbols');  xlabel('Real'),ylabel('Image')
  axis([-2 2 -2 2])
```

```
    subplot(222),
    plot(x,'o'); % show the pattern of received samples
    grid, title('Received samples'); xlabel('Real'), ylabel('Image')
    subplot(223),
    plot(sb,'o'); % show the pattern of the equalized symbols
    grid, title('Equalized symbols'), xlabel('Real'), ylabel('Image')
    subplot(224),
    plot(abs(e)); % show the convergence
    grid, title('Convergence'), xlabel('n'), ylabel('Error e(n)')
end
Z(6,:) = abs(e);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure
    subplot(321),
    plot(Z(1,:)); % show the convergence
    grid, title('Convergence P 1'), xlabel('n'), ylabel('Error e(n)')
    subplot(322),
    plot(Z(2,:)); % show the convergence
    grid, title('Convergence P 2'), xlabel('n'), ylabel('Error e(n)')
    subplot(323),
    plot(Z(3,:)); % show the convergence
    grid, title('Convergence P 3'), xlabel('n'), ylabel('Error e(n)')
    subplot(324),
    plot(Z(4,:)); % show the convergence
    grid, title('Convergence P 4'), xlabel('n'), ylabel('Error e(n)')
    subplot(325),
    plot(Z(5,:)); % show the convergence
    grid, title('Convergence P 5'), xlabel('n'), ylabel('Error e(n)')
    subplot(326),
    plot(Z(6,:)); % show the convergence
    grid, title('Convergence P 6'), xlabel('n'), ylabel('Error e(n)')
```

A.2. Simulation of SER and MSE verses SNR

```
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%SER / MSE verses SNR plot equation 2 in the paper and P = 5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
T=8000;   % total number of data
dB=29;    % SNR in dB value
%%%%%%%%% Simulate the Received noisy Signal  %%%%%%%%%%%
N=20; % smoothing length N+1
Lh=5;  % channel length = Lh+1
% for Channel-II use
%Lh=6;  % channel length = Lh+1
P=round((N+Lh)/2);  % equalization delay

%Channel - I shown in figure 4,1 (complex)
j=sqrt(-1);
h=[0.0545+j*0.05 .2832-.1197*j -.7676+.2788*j -.0641-.0576*j ...
     .0566-.2275*j .4063-.0739*j];
%Channel - II shown in figure 3.2 (complex)
%h=[-0.005-j*0.004 .009+.03*j -.024-.104*j .854+.520*j ...
%      -.218+.273*j .049-.074*j -.016+.20*j];
h=h/norm(h);                 % normalize

s=round(rand(1,T))*2-1;  % QPSK or 4 QAM symbol sequence
s=s+sqrt(-1)*(round(rand(1,T))*2-1);

ind=1; %array index
for dB=40:-1.25:5
   % generate received noisy signal
   x=filter(h,1,s);
   vn=randn(1,T)+sqrt(-1)*randn(1,T);  % AWGN noise (complex)
   vn=vn/norm(vn)*10^(-dB/20)*norm(x);  % adjust noise power with SNR dB value
   SNR=20*log10(norm(x)/norm(vn));      % Check SNR of the received samples
   x=x+vn;                        % received signal

   %%%%%%%%%%%% adaptive equalizer estimation via CMA
   Lp=T-N;  %% remove several first samples to avoid 0 or negative subscript
   X=zeros(N+1,Lp);  % sample vectors (each column is a sample vector)
   for i=1:Lp
      X(:,i)=x(i+N:-1:i).';
   end

   sb=zeros(1,Lp);
   e=zeros(1,Lp);  % used to save instant error
   f=zeros(N+1,1);
   % initial condition
   %for P = 5
   mu=0.0000001   ;     % parameter to adjust convergence and steady error
   R2=3.07;              % constant modulas of QPSK symbols

   for i=1:Lp
      %for P = 5
      e(i)=abs(f'*X(:,i))^5-R2;                % instant error
      f=f-mu*e(i)*2*X(:,i)*X(:,i)'*f .* abs(f'*X(:,i))^3 ;  % update equalizer
      f(P)=1;
   end
```

```
sb=f'*X;  % estimate symbols (perform equalization)

% calculate SER
H=zeros(N+1,N+Lh+1);
for i=1:N+1,
   H(i,i:i+Lh)=h;
end  % channel matrix

fh=f'*H ;  % composite channel+equalizer response should be delta-like
temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response

sb1=sb/(fh(temp));  % scale the output
sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
start=6;  % carefully find the corresponding begining point
sb2=sb1-s(start+1:start+length(sb1));  % find error symbols
SER=length(find(sb2~=0))/length(sb2);  % calculate SER
ser_arr_2(ind,:) = SER;

% calculate MSE
sb4=s(1:Lp);
Symbol_MSE=norm(sb-sb4)/Lp;  % estimation error MSE
mse_arr_2(ind,:) =Symbol_MSE;
ind=ind+1; %array index
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%SER  / MSE verses SNR plot equation 2.16 amd p = 5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
ind=1;%array index
for dB=40:-1.25:5 % SNR in dB value

  x=filter(h,1,s);
  vn=randn(1,T)+sqrt(-1)*randn(1,T);  % AWGN noise (complex)
  vn=vn/norm(vn)*10^(-dB/20)*norm(x);  % adjust noise power with SNR dB value
  SNR=20*log10(norm(x)/norm(vn));      % Check SNR of the received samples
  x=x+vn;                    % received signal

  %%%%%%%%%%%%%% adaptive equalizer estimation via CMA
  Lp=T-N;  %% remove several first samples to avoid 0 or negative subscript
  X=zeros(N+1,Lp);  % sample vectors (each column is a sample vector)
  for i=1:Lp
     X(:,i)=x(i+N:-1:i).';
  end
  sb=zeros(1,Lp);
  e=zeros(1,Lp);  % used to save instant error
  f=zeros(N+1,1);
  f(P)=1;   % initial condition
  mu=0.0000009;     % parameter to adjust convergence and steady error
  R2=1.20;

  for i=1:Lp
     %for P = 5
     e(i)=abs(f'*X(:,i))^5-R2;              % instant error
     f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;   % update equalizer
     f(P)=1;
  end
  sb=f'*X;  % estimate symbols (perform equalization)
  % calculate SER
  H=zeros(N+1,N+Lh+1);
```

```
  for i=1:N+1,
    H(i,i:i+Lh)=h;
  end  % channel matrix

  fh=f'*H ;  % composite channel+equalizer response should be delta-like
  temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response
  sb1=sb/(fh(temp));  % scale the output
  sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
  start=6 ;  % carefully find the corresponding begining point
  sb2=sb1-s(start+1:start+length(sb1));  % find error symbols
  SER=length(find(sb2~=0))/length(sb2);  % calculate SER

  ser_arr_11(ind,:) = SER;

  % calculate MSE
  sb4=s(1:Lp);
  Symbol_MSE=norm(sb-sb4)/Lp;  % estimation error MSE
  mse_arr_11(ind,:) =Symbol_MSE;
 ind=ind+1; %mse_arr_11 array index
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%SER  / MSE verses SNR plot for Eq. 2.16 for Multi Channel and P=5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Simulate the Received noisy Signal  %%%%%%%%%%%%
N=5; % smoothing length N+1
Lh=5;  % channel length = Lh+1
Ap=4;  % number of subchannels or receive antennas

h=randn(Ap,Lh+1)+sqrt(-1)*randn(Ap,Lh+1);  % channel (complex)
for i=1:Ap, h(i,:)=h(i,:)/norm(h(i,:));  end       % normalize
h=h/norm(h);
ind=1;
for dB=40:-1.25:5
  % generate received noisy signal
  x=zeros(Ap,T);   % matrix to store samples from Ap antennas
  SNR=zeros(1,Ap);
  for i=1:Ap
    x(i,:)=filter(h(i,:),1,s);
    vn=randn(1,T)+sqrt(-1)*randn(1,T);  % AWGN noise (complex)
    vn=vn/norm(vn)*10^(-dB/20)*norm(x(i,:));  % adjust noise power
    SNR(i)=20*log10(norm(x(i,:))/norm(vn));  % Check SNR of the received samples
    x(i,:)=x(i,:)+vn;                  % received signal
  end
  SNR=SNR;   % display and check SNR

  %%%%%%%%%%%% adaptive equalizer estimation via CMA
  Lp=T-N;  %% remove several first samples to avoid 0 or negative subscript
  X=zeros((N+1)*Ap,Lp);  % sample vectors (each column is a sample vector)
  for i=1:Lp
    for j=1:Ap
      X((j-1)*(N+1)+1:j*(N+1),i)=x(j, i+N:-1:i).';
    end
  end

  %sb=zeros(1,Lp);
  e=zeros(1,Lp);  % used to save instant error
  f=zeros((N+1)*Ap,1); %f(N*Ap/2)=1;   % initial condition
  %for P = 5
  mu=0.0000001;     % parameter to adjust convergence and steady error
  R2=2.5;
```

```
  for i=1:Lp
      %for P = 5
      e(i)=abs(f'*X(:,i))^5-R2;              % instant error
      f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
      f(N*Ap/2)=1;
   end

  sb=f'*X;  % estimate symbols (perform equalization)
  % calculate SER
  H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
  for j=1:Ap
     for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
  end
  fh=f'*H;   % composite channel+equalizer response should be delta-like
  temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response
  sb1=sb/(fh(temp));  % scale the output
  sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
  start=N+1-temp;  % general expression for the beginning matching point
  sb2=sb1(10:length(sb1))-s(start+10:start+length(sb1));  % find error symbols
  SER=length(find(sb2~=0))/length(sb2);  % calculate SER
  ser_arr_mul(ind,:) = SER;

  % calculate MSE
  sb4=s(1:Lp);
  Symbol_MSE=norm(sb4-sb)/Lp;  % estimation error MSE
  mse_arr_mul(ind,:) =Symbol_MSE;
  ind=ind+1; %mse_arr_11 array index
end
i=1;
for SNR=40:-1.25:5
  snr_arr(i,:)=SNR;
  i=i+1;
end
figure
     plot(snr_arr, ser_arr_2,'-b+');  % show the convergence
     grid, title('SNR / SER Plot, Comparision'), xlabel('SNR'), ylabel('SER')
     hold on
     plot(snr_arr,ser_arr_11,'-gd');  % show the convergence
     %plot(snr_arr,ser_arr_mul,'-rx');  % show the convergence
     legend('EQ 2','EQ 11','Multi Channel')
 figure
     plot(snr_arr, mse_arr_2,'-b');  % show the convergence
     grid, title('SNR / MSE Plot, Comparision'), xlabel('SNR'), ylabel('MSE')
     hold on
     plot(snr_arr,mse_arr_11,'-g');  % show the convergence
     plot(snr_arr,mse_arr_mul,'-r');  % show the convergence
     legend('EQ 2','EQ 11','Multi Channel')
```

A.3. Simulation SNR, MSE & NRMS ISI verses No of Cycles (EPOC)

```
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%SER / MSE / ISI verses no of cycles (EPOC) using Equation 3.9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% adpative CMA method
T=8000;   % total number of data
dB=25;    % SNR in dB value
%%%%%%%%%% Simulate the Received noisy Signal  %%%%%%%%%%%
N=20; % smoothing length N+1
Lh=5;  % channel length = Lh+1

% for Channel-II use
%Lh=6;  % channel length = Lh+1

P=round((N+Lh)/2);  % equalization delay

% Channel-I (complex) shown in figure 3.1
j=sqrt(-1);
h=[0.0545+j*0.05 .2832-.1197*j -.7676+.2788*j -.0641-.0576*j ...
     .0566-.2275*j .4063-.0739*j];

%Channel - II shown in figure 3.2 (complex)
%h=[-0.005-j*0.004 .009+.03*j -.024-.104*j .854+.520*j ...
%      -.218+.273*j .049-.074*j -.016+.20*j];

h=h/norm(h);                % normalize
s=round(rand(1,T))*2-1;  % QPSK or 4 QAM symbol sequence
s=s+sqrt(-1)*(round(rand(1,T))*2-1);

% generate received noisy signal
x=filter(h,1,s);
vn=randn(1,T)+sqrt(-1)*randn(1,T);  % AWGN noise (complex)
vn=vn/norm(vn)*10^(-dB/20)*norm(x);  % adjust noise power with SNR dB value
SNR=20*log10(norm(x)/norm(vn))     % Check SNR of the received samples
x=x+vn;                    % received signal

%%%%%%%%%%%%% adaptive equalizer estimation via CMA
Lp=T-N;  %% remove several first samples to avoid 0 or negative subscript
X=zeros(N+1,Lp);  % sample vectors (each column is a sample vector)
for i=1:Lp
   X(:,i)=x(i+N:-1:i).';
end

e=zeros(1,Lp);  % used to save instant error
f=zeros(N+1,1); f(P)=1;   % initial condition

%for P = 2
mu=0.00000019;     % parameter to adjust convergence and steady error
R2=3.1;            % constant modulas of QPSK symbols
MAX=200 % the max numbers of cycles or Epoc
mse_arr_2=zeros(MAX,1);
ser_arr_2=zeros(MAX,1);
nrmisi_2=zeros(MAX,1);

for epoc=1:MAX
```

```
  for i=1:Lp
     %for P = 5
     e(i)=abs(f'*X(:,i))^5-R2;                % instant error
     f=f-mu*e(i)*2*X(:,i)*X(:,i)'*f .* abs(f'*X(:,i))^3 ;  % update equalizer
     f(P)=1;
  end

  sb=f'*X;  % estimate symbols (perform equalization)

  % calculate SER
  H=zeros(N+1,N+Lh+1);
  for i=1:N+1,
     H(i,i:i+Lh)=h;
  end  % channel matrix1

  fh=f'*H ;  % composite channel+equalizer response should be delta-like
  temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response

  sb1=sb/(fh(temp));  % scale the output
  sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
  start=6;  % carefully find the corresponding begining point
  sb2=sb1-s(start+1:start+length(sb1));  % find error symbols
  SER=length(find(sb2~=0))/length(sb2);   % calculate SER
  ser_arr_2(epoc,:) = SER;

  % calculate MSE
  sb4=s(1:Lp);
  Symbol_MSE=norm(sb-sb4)/Lp;  % estimation error MSE
  mse_arr_2(epoc,:) =Symbol_MSE;

  % calculate ISI
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
     d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
     isi = isi + abs(sb4(j) - d*sb(j))^2;
  end

  nrmisi_2(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%SER / MSE / ISI verses no of cycles (EPOC) using Equation 2.16
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%sb=zeros(1,Lp);  %
e=zeros(1,Lp);  % used to save instant error
f=zeros(N+1,1); f(P)=1;    % initial condition

%for P = 5
mu=0.0000011;     % parameter to adjust convergence and steady error
R2=1.75;              % constant modulas of QPSK symbols
mse_arr_11=zeros(MAX,1);
ser_arr_11=zeros(MAX,1);
nrmisi_11=zeros(MAX,1);

for epoc=1:MAX
   epoc
   for i=1:Lp
      %for P = 5
```

```
    e(i)=abs(f'*X(:,i))^5-R2;              % instant error
    f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
    f(P)=1;
  end

  sb=f'*X;  % estimate symbols (perform equalization)

  % calculate SER
  H=zeros(N+1,N+Lh+1);
  for i=1:N+1,
    H(i,i:i+Lh)=h;
  end  % channel matrix

  fh=f'*H ;  % composite channel+equalizer response should be delta-like
  temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response

  sb1=sb/(fh(temp));  % scale the output
  sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
  start=6 ;  % carefully find the corresponding begining point
  sb2=sb1-s(start+1:start+length(sb1));  % find error symbols
  SER=length(find(sb2~=0))/length(sb2);  % calculate SER
  ser_arr_11(epoc,:) = SER;

  % calculate MSE
  sb4=s(1:Lp);
  Symbol_MSE=norm(sb-sb4)/Lp;  % estimation error MSE
  mse_arr_11(epoc,:) =Symbol_MSE;
  % calculate ISI
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
     d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
     isi = isi + abs(sb4(j) - d*sb(j))^2;
  end
  nrmisi_11(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%SER/ MSE/ ISI verses no of cycles (EPOC) using multiple channelEquation 2.16
%%%%%%%% Simulate the Received noisy Signal  %%%%%%%%%%%
N=5; % smoothing length N+1
Lh=5;  % channel length = Lh+1
Ap=4;  % number of subchannels or receive antennas

h=randn(Ap,Lh+1)+sqrt(-1)*randn(Ap,Lh+1);  % channel (complex)
for i=1:Ap, h(i,:)=h(i,:)/norm(h(i,:));  end      % normalize

% generate received noisy signal
x=zeros(Ap,T);   % matrix to store samples from Ap antennas
SNR=zeros(1,Ap);
for i=1:Ap
  x(i,:)=filter(h(i,:),1,s);
  vn=randn(1,T)+sqrt(-1)*randn(1,T);  % AWGN noise (complex)
  vn=vn/norm(vn)*10^(-dB/20)*norm(x(i,:));  % adjust noise power
  SNR(i)=20*log10(norm(x(i,:))/norm(vn));  % Check SNR of the received samples
  x(i,:)=x(i,:)+vn;                    % received signal
end
SNR=SNR   % display and check SNR
%%%%%%%%%%%%% adaptive equalizer estimation via CMA
```

```
Lp=T-N;  %% remove several first samples to avoid 0 or negative subscript
X=zeros((N+1)*Ap,Lp);  % sample vectors (each column is a sample vector)
for i=1:Lp
  for j=1:Ap
    X((j-1)*(N+1)+1:j*(N+1),i)=x(j, i+N:-1:i).';
  end
end
e=zeros(1,Lp);  % used to save instant error
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
%for P = 5
mu=0.00000029;     % parameter to adjust convergence and steady error
R2=0.45;           % constant modulas of QPSK symbols
mse_arr_mul=zeros(MAX,1);
ser_arr_mul=zeros(MAX,1);
nrmisi_mul=zeros(MAX,1);

for epoc=1:MAX
  epoc
  for i=1:Lp
    %for P = 5
    e(i)=abs(f'*X(:,i))^5-R2;              % instant error
    f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
    f(N*Ap/2)=1;
  end

  sb=f'*X;  % estimate symbols (perform equalization)
  % calculate SER
  H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
  for j=1:Ap
    for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
  end
  fh=f'*H;   % composite channel+equalizer response should be delta-like
  temp=find(abs(fh)==max(abs(fh)));  % find the max of the composite response
  sb1=sb/(fh(temp));  % scale the output
  sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));  % perform symbol detection
  start=N+1-temp;  % general expression for the beginning matching point
  sb2=sb1(10:length(sb1))-s(start+10:start+length(sb1));  % find error symbols
  SER=length(find(sb2~=0))/length(sb2);  % calculate SER
  ser_arr_mul(epoc,:) = SER;

  % calculate MSE
  sb4=s(1:Lp);
  Symbol_MSE=norm(sb-sb4)/Lp;  % estimation error MSE
  mse_arr_mul(epoc,:) =Symbol_MSE;

  % calculate ISI
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
    d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
    isi = isi + abs(sb4(j) - d*sb(j))^2;
  end
  nrmisi_mul(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
    figure
    plot(ser_arr_mul,'-.r');
    grid, title('Number of Cycles / MSE Plot, Comparison'), xlabel('No of Cycles'), ylabel('SER')
    hold on
```

```
plot(ser_arr_11,':g');
plot(ser_arr_2,'-b');
legend('Multi Channel','Equation 11','Equation 2' )

figure
plot(mse_arr_mul,'-.r');
grid, title('Number of Cycles / MSE Plot, Comparison'), xlabel('No of Cycles'), ylabel('SER')
hold on
plot(mse_arr_11,':g');
plot(mse_arr_2,'-b');
legend('Multi Channel','Equation 11','Equation 2' )

figure
plot(nrmisi_2,'-b');
grid, title('Number of Cycles / ISI Plot, Comparison'), xlabel('No of Cycles'), ylabel('ISI')
hold on
plot(nrmisi_11,'-g');
plot(nrmisi_mul,'r');
legend('Equation 2','Equation 11','Multi Channel' )
```

A.4. Calculation NRMS ISI for different values of Multipath (P) using Fractional Space Algorithm

```
clear all
clc
T=8000;   % total number of data
dB=25;    % SNR in dB value
%%%%%%%%%% Simulate the Received noisy Signal  %%%%%%%%%%%%
N=5; % smoothing length N+1
Lh=5;  % channel length = Lh+1
Ap=4;  % number of subchannels or receive antennas

h=randn(Ap,Lh+1)+sqrt(-1)*randn(Ap,Lh+1);  % channel (complex)
for i=1:Ap, h(i,:)=h(i,:)/norm(h(i,:));   end      % normalize

s=round(rand(1,T))*2-1;  % QPSK or 4 QAM symbol sequence
s=s+sqrt(-1)*(round(rand(1,T))*2-1);

% generate received noisy signal
x=zeros(Ap,T);   % matrix to store samples from Ap antennas
SNR=zeros(1,Ap);
for i=1:Ap
   x(i,:)=filter(h(i,:),1,s);
   vn=randn(1,T)+sqrt(-1)*randn(1,T);  % AWGN noise (complex)
   vn=vn/norm(vn)*10^(-dB/20)*norm(x(i,:));  % adjust noise power
   SNR(i)=20*log10(norm(x(i,:))/norm(vn));  % Check SNR of the received samples
   x(i,:)=x(i,:)+vn;                  % received signal
end
SNR=SNR   % display and check SNR
%%%%%%%%%%%%%% adaptive equalizer estimation via CMA
Lp=T-N;  %% remove several first samples to avoid 0 or negative subscript
X=zeros((N+1)*Ap,Lp);  % sample vectors (each column is a sample vector)
for i=1:Lp
   for j=1:Ap
      X((j-1)*(N+1)+1:j*(N+1),i)=x(j, i+N:-1:i).';
   end
end

e=zeros(1,Lp);  % used to save instant error
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
MAX=1000 % the max numbers of cycles or Epoc
%%%%%%%%%%%%%%%%%%%%%%%%%%  P = 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nrmisi_mul_1=zeros(MAX,1);
e=zeros(1,Lp);  % used to save instant error
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
%P = 1
mu=0.0000002;     % parameter to adjust convergence and steady error
R2=1.25;            % constant modulas of QPSK symbols
for epoc=1:MAX
   epoc
   for i=1:Lp
      %for P = 1
      e(i)=abs(f'*X(:,i))-R2;              % instant error
      f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
      f(N*Ap/2)=1;
   end
   sb=f'*X;  % estimate symbols (perform equalization)
   % calculate SER
```

```
  H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
  for j=1:Ap
    for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
  end
  fh=f'*H;   % composite channel+equalizer response should be delta-like
  %calculating ISI
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
    d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
    isi = isi + abs(sb4(j) - d*sb(j))^2;
  end
  nrmisi_mul_1(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
%%%%%%%%%%%%%%%%%%%%%%  P = 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nrmisi_mul_2=zeros(MAX,1);
e=zeros(1,Lp);  % used to save instant error
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
mu=0.00000003;     % parameter to adjust convergence and steady error
R2=1.25;           % constant modulas of QPSK symbols
for epoc=1:MAX
  epoc
  for i=1:Lp
    %for P = 1
    e(i)=abs(f'*X(:,i))^2-R2;              % instant error
    f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
    f(N*Ap/2)=1;
  end
  sb=f'*X;  % estimate symbols (perform equalization)
  % calculate SER
  H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
  for j=1:Ap
    for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
  end
  fh=f'*H;   % composite channel+equalizer response should be delta-like
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
    d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
    isi = isi + abs(sb4(j) - d*sb(j))^2;
  end
  nrmisi_mul_2(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
%%%%%%%%%%%%%%%%%%%%%%  P = 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nrmisi_mul_3=zeros(MAX,1);
e=zeros(1,Lp);  % used to save instant error
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
mu=0.00000003;     % parameter to adjust convergence and steady error
R2=1.45;           % constant modulas of QPSK symbols
for epoc=1:MAX
  epoc
  for i=1:Lp
    %for P = 3
    e(i)=abs(f'*X(:,i))^3-R2;              % instant error
```

```
    f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
    f(N*Ap/2)=1;
  end
  sb=f'*X;  % estimate symbols (perform equalization)
  % calculate SER
  H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
  for j=1:Ap
    for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
  end
  fh=f'*H;   % composite channel+equalizer response should be delta-like
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
    d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
    isi = isi + abs(sb4(j) - d*sb(j))^2;
  end
  nrmisi_mul_3(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
%%%%%%%%%%%%%%%%%%%%%%%  P = 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nrmisi_mul_4=zeros(MAX,1);
e=zeros(1,Lp);  % used to save instant error
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
mu=0.00000003;     % parameter to adjust convergence and steady error
R2=1.4;              % constant modulas of QPSK symbols
for epoc=1:MAX
  epoc
  for i=1:Lp
    %for P = 4
     e(i)=abs(f'*X(:,i))^4-R2;              % instant error
    f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
    f(N*Ap/2)=1;
  end
  sb=f'*X;  % estimate symbols (perform equalization)

  % calculate SER
  H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
  for j=1:Ap
    for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
  end
  fh=f'*H;   % composite channel+equalizer response should be delta-like
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
    d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
    isi = isi + abs(sb4(j) - d*sb(j))^2;
  end
  nrmisi_mul_4(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
%%%%%%%%%%%%%%%%%%%%%%%  P = 5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nrmisi_mul_5=zeros(MAX,1);
e=zeros(1,Lp);  % used to save instant error
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
mu=0.00000003;     % parameter to adjust convergence and steady error
R2=1.3;              % constant modulas of QPSK symbols
```

```
for epoc=1:MAX
  epoc
  for i=1:Lp
    %for P = 5
     e(i)=abs(f'*X(:,i))^5-R2;              % instant error
    f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
    f(N*Ap/2)=1;
  end
  sb=f'*X;  % estimate symbols (perform equalization)
  % calculate SER
  H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
  for j=1:Ap
    for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
  end
  fh=f'*H;   % composite channel+equalizer response should be delta-like
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
    d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
    isi = isi + abs(sb4(j) - d*sb(j))^2;
  end
  nrmisi_mul_5(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
%%%%%%%%%%%%%%%%%%%%%%%  P = 6
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nrmisi_mul_6=zeros(MAX,1);
e=zeros(1,Lp);  % used to save instant error
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
mu=0.00000003;     % parameter to adjust convergence and steady error
R2=1.15;              % constant modulas of QPSK symbols
for epoc=1:MAX
  epoc
  for i=1:Lp
     e(i)=abs(f'*X(:,i))^6-R2;              % instant error
    f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
    f(N*Ap/2)=1;
  end
  sb=f'*X;  % estimate symbols (perform equalization)
  % calculate SER
  H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
  for j=1:Ap
    for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
  end
  fh=f'*H;   % composite channel+equalizer response should be delta-like
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
    d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
    isi = isi + abs(sb4(j) - d*sb(j))^2;
  end
  nrmisi_mul_6(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
%%%%%%%%%%%%%%%%%%%%%%%  P = 7
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nrmisi_mul_7=zeros(MAX,1);
e=zeros(1,Lp);  % used to save instant error
```

```
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
mu=0.00000002;     % parameter to adjust convergence and steady error
R2=1.1;            % constant modulas of QPSK symbols
for epoc=1:MAX
  epoc
  for i=1:Lp
    %for P = 6
     e(i)=abs(f'*X(:,i))^7-R2;              % instant error
    f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
    f(N*Ap/2)=1;
  end
  sb=f'*X;  % estimate symbols (perform equalization)
  % calculate SER
  H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
  for j=1:Ap
    for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
  end
  fh=f'*H;   % composite channel+equalizer response should be delta-like
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
    d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
    isi = isi + abs(sb4(j) - d*sb(j))^2;
  end
  nrmisi_mul_7(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
%%%%%%%%%%%%%%%%%%%%%%  P = 8
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nrmisi_mul_8=zeros(MAX,1);
e=zeros(1,Lp);  % used to save instant error
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
mu=0.00000002;     % parameter to adjust convergence and steady error
R2=1.01;           % constant modulas of QPSK symbols
for epoc=1:MAX
  epoc
  for i=1:Lp
    %for P = 6
     e(i)=abs(f'*X(:,i))^8-R2;              % instant error
    f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
    f(N*Ap/2)=1;
  end
  sb=f'*X;  % estimate symbols (perform equalization)
  % calculate SER
  H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
  for j=1:Ap
    for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
  end
  fh=f'*H;   % composite channel+equalizer response should be delta-like
  sb4=s(1:Lp);
  Nm=MAX; %number of monte carlo trials
  alpha=1; %rotational constant
  isi=0;
  for j=1:Lp
    d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
    isi = isi + abs(sb4(j) - d*sb(j))^2;
  end
  nrmisi_mul_8(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%  P = 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nrmisi_mul_9=zeros(MAX,1);
e=zeros(1,Lp);  % used to save instant error
f=zeros((N+1)*Ap,1); f(N*Ap/2)=1;   % initial condition
mu=0.00000001;     % parameter to adjust convergence and steady error
R2=1.001;              % constant modulas of QPSK symbols
for epoc=1:MAX
   epoc
   for i=1:Lp
      %for P = 6
       e(i)=abs(f'*X(:,i))^9-R2;              % instant error
      f=f-mu*2*e(i)*X(:,i)*X(:,i)'*f;    % update equalizer
      f(N*Ap/2)=1;
   end
   sb=f'*X;  % estimate symbols (perform equalization)
   % calculate SER
   H=zeros((N+1)*Ap,N+Lh+1);  temp=0;
   for j=1:Ap
      for i=1:N+1, temp=temp+1; H(temp,i:i+Lh)=h(j,:); end  % channel matrix
   end
   fh=f'*H;   % composite channel+equalizer response should be delta-like
   sb4=s(1:Lp);
   Nm=MAX; %number of monte carlo trials
   alpha=1; %rotational constant
   isi=0;
   for j=1:Lp
      d=sqrt ((real(sb(j))-real(sb4(j)) )^2+ (imag(sb(j))-imag(sb4(j)) )^2);
      isi = isi + abs(sb4(j) - d*sb(j))^2;
   end
   nrmisi_mul_9(epoc)=1/norm(s)*sqrt(1/Nm*isi);
end
   figure
    plot(nrmisi_mul_1,'-b');
    grid, title('Number of Cycles / ISI Plot, P Comparison EQ Mul'), xlabel('No of Cycles'), ylabel('ISI')
    hold on
    plot(nrmisi_mul_2,'-g');
    plot(nrmisi_mul_3,'r');
    plot(nrmisi_mul_4,'-c');
    plot(nrmisi_mul_5,'m');
    plot(nrmisi_mul_6,'c');
    plot(nrmisi_mul_7,'k');
    plot(nrmisi_mul_8,'-.b');
    plot(nrmisi_mul_9,'-.g');
    legend('P 1','P 2','P 3','P 4','P 5','P 6','P 7','P 8','P 9'
```

References

[1] Godard, D.N. 'Self-recovering equalization and carrier tracking in two dimensional data communication systems', IEEE Trans. Commun., 1980, COM-28, (Il), pp. 1867-1875

[2] Simon Haykin, Adaptive Filter Theory, 4nd Ed., .Upper Saddle River.NJ:Prentice Hall,2002

[3] Sato, Y 'A method of self-recovering equalization for multilevel amplitude modulation systems', IEEE Trans. Commun., 1975,

[4] Treichler, J.R., and Acee, B.G.: 'A new approach to multipath correction of constant modulus signals', IEEE Trans. Acoust. Speech Signd Process, 1983, ASSP-31, (2), pp. 459472

[5] Treichler, J.R., and Larimore, M.G.: 'New processing techniques based on the constant modulus adaptive algorithm', IEEE Trans. Acoust. Speech Signal Process, 1985, ASSP-33, (2), pp. 420-431

[6] Picchi, C., and Prati, C. 'Blind equalization and carrier recovery using a stop-and-go decision-directed algorithm', IEEE Trans. Commun., 1987, COM-35, (9), pp. 877-887

[7] Weerackody, V., Kassam, S.A., and Laker, K.R.: 'Blind equalisation using lattice filters'. Proceedings of the International Conference on Communication, ICC'98, 1988

[8] Bellini, S.: 'Bussgang techniques for blind equalisation'. Proceedings of Globecom, 1986, pp. 1634-1640

[9] White, L.B.: 'Blind equalization of constant modulus signals using an adaptive observer approach', IEEE Truns. Commun., 1996, 44, (2), pp. 134-136

[10] K. N. Oh and Y. O. Chin, "Modified constant modulus algorithm: Blind equalization and carrier phase recovery algorithm," in Proc. IEEE ICC, 1995, vol. 1, pp. 498–502.

[11] Shafat Abrar and Syed Ismail Shah, "New Multimodulus Blind Equalization Algorithm With Relaxation" in Proc. IEEE ICM', 2005

[12] J.-C. Lin, "Blind equalization technique based on an improved constant modulus adaptive algorithm," Proc. Inst. Elect. Eng., Commun., vol. 149, no. 1, pp. 45–50, 2002.

[13] S. Abrar, A. Zerguine, and M. Deriche, "Soft constraint satisfaction multimodulus blind equalization algorithms," IEEE Signal Processing Lett., vol. 12, no. 9, pp. 637–640, 2005.

[14] J. Mai and A. H. Sayed, "A feedback approach to the steady-state performance of fractionally-spaced blind adaptive equalizers," IEEE Trans. Signal Process. vol. 48, no. 1, pp. 80–91, 2000.