A project reported submitted in partial fulfillment of
the requirements of the course EEL6503:
Spread Spectrum and CDMA, Fall 2001

Submitted by
Arun Avudainaygam

Project Website: http://arun-10.tripod.com/mud/mud.html

LINEAR AND ADAPTIVE LINEAR MULTIUSER DETECTION IN CDMA SYSTEMS

SECTION 0

Introduction

Multiuser detection is a technology that spawned in the early 80's. It has now developed into an important, full-fledged field in multi-access communications.

Multiuser Detection (MUD) is the intelligent estimation/demodulation of transmitted bits in the presence of Multiple Access Interference (MAI). MAI occurs in multi-access communication systems (CDMA/ TDMA/ FDMA) where simultaneously occurring digital streams of information interfere with each other. Conventional detectors based on the matched filter just treat the MAI as additive white gaussian noise (AWGN). However, unlike AWGN, MAI has a nice correlative structure that is quantified by the cross-correlation matrix of the signature sequences. Hence, detectors that take into account this correlation would perform better than the conventional matched filter-bank. MUD is basically the design of signal processing algorithms that run in the black box shown in figure 0.1. These algorithms take into account the correlative structure of the MAI.

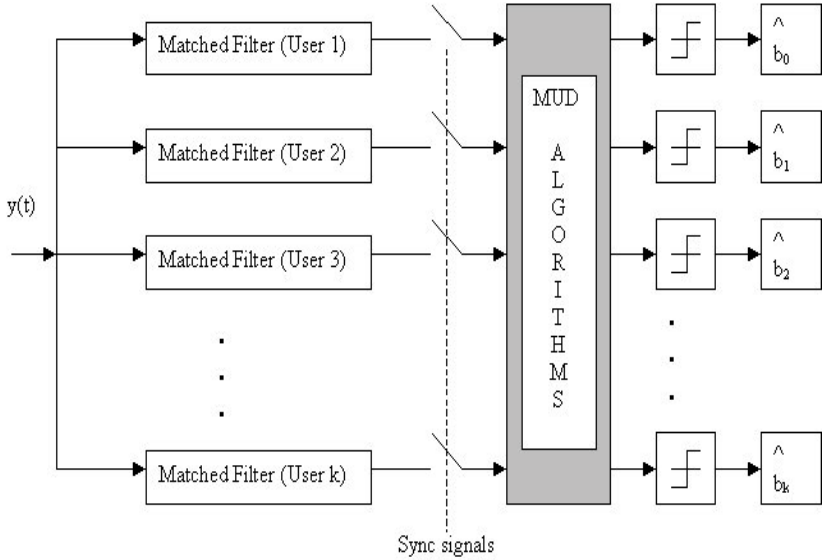

Figure 0.1 A typical multiuser detector

0.1 Overview of the project

This project investigates a couple of different approaches to linear multiuser detection in CDMA systems. Linear MUDs are detectors that operate linearly on the received signal statistic i.e., they perform only linear transformations on the received statistic.

This report assumes that the reader has a basic knowledge of probability theory and random processes and is familiar with the fundamental concepts of spread spectrum and CDMA systems.

0.2 Organization of the report

This report has been typeset in the 2-column landscape format to help preserve continuity in the mathematical development in some sections. The reader need not keep flipping pages to keep up with the large number of equations that are involved in certain sections. The report is orgainzed as follows:

- The next sub-section introduces the system model that will be used throught this project.
- Section I presents the conventional method of demodulating mutually interfering signals: the matched filter (which treats the MAI as AWGN).
- Section II studies the decorrelating detector which takes the matched filter one step further by taking into account the correlative structure of the MAI.
- Section III presents the minimum mean square error (MMSE) linear detector which is a compromise between the matched filter approach and the decorrelating detector. Two different adaptive implementations (least mean square or LMS and blind adaptation) are implemented and the convergence properties are studied.

0.3 System model

The K-user discrete time basic synchronous CDMA model has been used throughout the development of this project. The case of antipodally modulated user information (BPSK modulation) spread using BPSK spreading is considered. To make the project realizable in the time allocated, a very small spreading sequence of length 31 was used. A preferred pair [1,2] of m-sequences generated by the primitive polynomials <45> and <75> were used for all the 2-user scenarios. For the number of users greater than 2, gold codes generated by the 2 m-sequences

described above were used. See Appendix A for a list of codes in this project. Unless otherwise mentioned, in all the simulations perfect power control is assumed (i.e, the received amplitudes of all the users are assumed to be the same. The signal at the receiver is given by

$$y(t) = \sum_{k=1}^{K} A_k b_k s_k(t) + n(t) \quad \text{.....(0.1)}$$

, where

- s_k is the signature waveform of the k^{th} user (s_k is normalized to have unit energy i.e., $< s_k , s_k > = 1$). For BPSK spreading with an m-sequence of length 31, the signature waveform is defined as

  $s_k(t) = \sum_{k=0}^{30} a_k p_T(t - kT_c)$, T=bit period, T_c =chip interval… (0.2)

  where a_k represents the normalized spreading sequence.
- b_k is the input bit of the k^{th} user, $b_k \in \{-1,1\}$.
- A_k is the received amplitude of the k^{th} user.
- $n(t)$ is additive white gaussian noise with PSD No .

Since synchronous CDMA is considered, it is assumed that the receiver has some means of achieving perfect chip synchronization.

The cross-correlation of the signature sequences are defined as

$$\rho_{ij} = \langle s_i s_j \rangle = \sum_{k=1}^{N} s_i(k) s_j(k) \quad \text{.....(0.3)}$$

where N is the length of the signature sequence (31 in our case).

The cross-correlation matrix is then defined as

$$\mathbf{R} = \{\rho_{ij}\}$$

i.e., $$\mathbf{R} = \begin{bmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1K} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{21K} \\ \vdots & \vdots & \cdots & \vdots \\ \rho_{K1} & \rho_{K2} & \cdots & \rho_{KK} \end{bmatrix} \quad \text{.....(0.4)}$$

$\mathbf{R}$ is a symmetric, non-negative definite, toeplitz matrix.

SECTION I

The Matched-Filter Bank

This section introduces and analyses the matched filter bank detector which was the conventional and most simplest way of demodulating CDMA signals (or any other set of mutually interfering digital streams). The matched filter also forms the front-end in most MUDs and hence understanding the operation is crucial in appreciating the evolution of MUD technology.

2.0 Receiver Operation

In conventional single-user digital communication systems, the matched filter is used to generate sufficient statistics for signal detection. In the case of a multi-user system, the detector consists of a bank of matched filters (each matched to the signature waveforms of different users in the case of CDMA). This is shown in figure 1.1. This type of detector is referred to as the *conventional detector* in MUD literature. It is worth mentioning that we need exact knowledge of the users signature sequences and the signal timing in order to implement this detector.

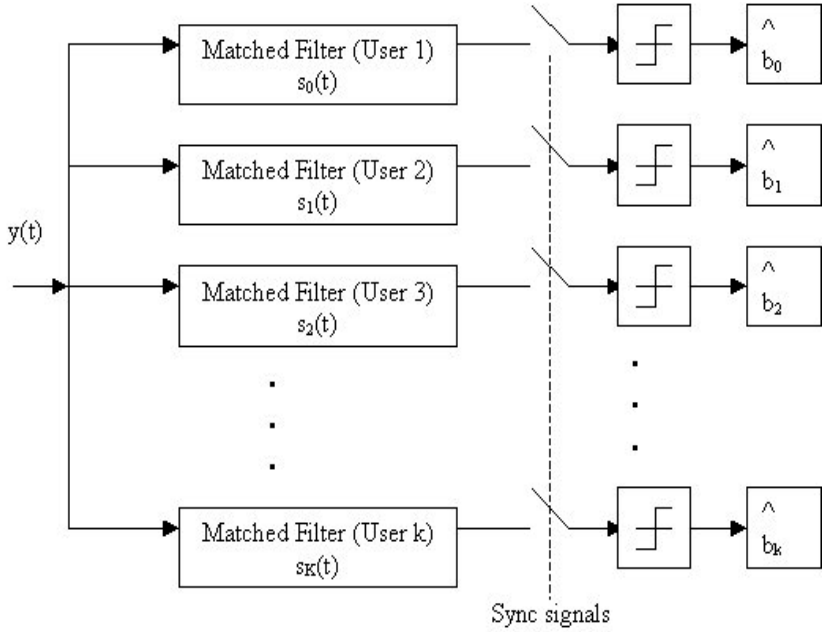

Figure 1.2 A matched filter bank

The decision statistic a the output of the K^{th} matched filter is given by

$$y_k = \int_0^T y(t)s_k(t)dt \quad \text{.....(2.1)}$$

where y(t) and $s_k(t)$ is given by (0.1) and (0.2). Expanding (2.1),

$$y_k = \int_0^T \left\{ \sum_{j=1}^{K} A_j b_j s_j(t) + n(t) \right\} s_k(t)dt \quad \text{.....(2.2)}$$

Using (0.3)

$$y_k = \sum_{j=1}^{K} A_j b_j \rho_{jk} + n_k \quad \text{.....(2.3)}$$

where $$n_k = \int_0^T n(t)s_k(t)dt \quad \text{.....(2.4)}$$

Since $\rho_{11}=1$, (2.3) simplifies to

$$y_k = A_k b_k + \sum_{\substack{j=1 \\ j\neq k}}^{K} A_j b_j \rho_{jk} + n_k \quad \text{.....(2.5)}$$

The 2nd term in (2.5) is the MAI. The matched filter treats the MAI just as white noise. The noise variance at the output of the matched filter is given by

$$E(n_k^2) = E\left[\int_0^T n(t)s_k(t)dt \int_0^T n(s)s_k(s)ds\right] = \int_0^T \int_0^T E[n(t)n(s)]s_k(s)s_k(t)dtds$$

$$= \int_0^T \int_0^T N_o \delta(t-s)s_k(s)s_k(t)dtds = \int_0^T N_o s_k^2(t)dt = N_o \quad \text{...(2.6)}$$

Similarly, the noise covariance can be shown to be

$$E(n_i n_j) = N_o \rho_{ij} \quad \text{.....(2.7)}$$

Hence the noise covariance matrix can be defined as

$$E[\mathbf{n}\mathbf{n}^T] = \{N_o \rho_{ij}\}_{ij} = N_o \mathbf{R} \quad \text{.....(2.8)}$$

where $\mathbf{R}$ is given by (0.4) and $\mathbf{n}=[n_1, n_2, \ldots, n_k]^T$. Stacking up (2.5) for all the users we get

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix} = \begin{bmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1K} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{21K} \\ \vdots & \vdots & \cdots & \vdots \\ \rho_{K1} & \rho_{K2} & \cdots & \rho_{KK} \end{bmatrix} \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & A_K \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_K \end{bmatrix}$$

…….....(2.9)

∴ In matrix notation we have,

$$\mathbf{y}=\mathbf{RAb}+\mathbf{n} \quad \text{……(2.10)}$$

Figure 2.2 shows the error rate performance of the bank of matched filters. The simulation scenario is explained in section 0.3. It is observed that as the MAI increases (the number of users increases) the performance becomes poor. This is because the detector ignores the cross-talk between users (the MAI) as white noise. Good MUDs, as described in the next few sections, take into the account the correlative property of the cross-talk.

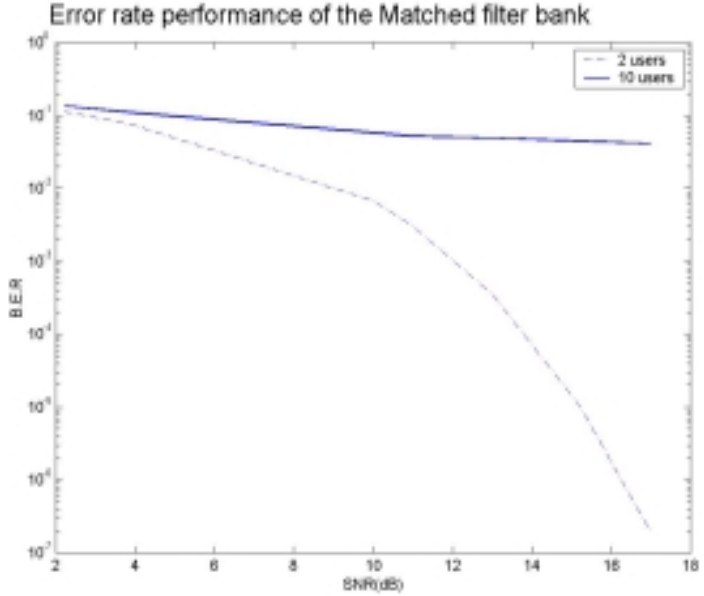

Figure 1.2 B.E.R performance of the matched filter bank detector (perfect power control)

2.1 Limitations of the conventional detector

Although $\{y_1,y_2,\ldots,y_k\}$ are sufficient statistics for detecting $\{b_1,b_2,\ldots,b_k\}$, y_k is not a sufficient statistic for detecting b_k. The conventional detector makes the mistake of making this assumption (y_k is a sufficient statistic for detecting b_k) by ignoring the MAI as background noise. This is one reason for the poor performance of the matched filter bank when the number of users is large.

Another serious limitation of the conventional detector is that it is seriously affected by the near-far problem. This causes a significant degradation in the system performance even when the number of users is very small. This fact will now be illustrated with an example. This example is a condensed version of a scenario described in [3]. Adapting (2.5) to the 2 user scenario we get,

$$y_1 = A_1 b_1 + A_2 b_{2j} \rho_{12} + n_1 \quad \text{……..(2.11)}$$

It is now obvious that the bit error probability for user 1 is

$$P_{s_1} = \frac{1}{2}\left\{ Q\left(\frac{A_1 + A_2\rho_{12}}{N_o} \right) + Q\left(\frac{A_1 - A_2\rho_{12}}{N_o} \right) \right\} \quad \text{……(2.12)}$$

The probability of bit error is then readily upper bounded as

$$P_{s_1} \le \frac{1}{2} Q\left(\frac{A_1 - A_2|\rho_{12}|}{N_o} \right) \quad \text{……..……(2.13)}$$

The fact that Q is a monotonically decreasing function was used to get the upper bound. If the interferer is not dominant i.e., $A_2\rho_{12}<A_1$, the bit error probability is less than half. But if the interferer is dominant (near-far problem) i.e., $A_2\rho_{12}>A_1$, the bound becomes greater than half. Consider the case when there is no noise in the system (i.e., $N_o=0$) and the interferer is dominant, then (2,13) gives $P_{s_1} = \frac{1}{2}$. This is because the polarity of the matched filter outputs is now governed by b_2 rather than b_1. Hence we see that in the absence of noise, though highly hypothetical, the matched filter receiver reduces to flipping a coin and deciding the output bits. This is an undesirable feature of the conventional detector (may perform better in the presence of noise than in the absence of noise).

2.2 The conventional detector as a front end to MUDs

The front end of any MUD has a section to convert the continuous-time received signal to a discrete-time process. This is usually done by sampling or it can also be done using the matched filter bank. As shown earlier, the conventional detector takes the received signal y(t) and outputs the statistic $\mathbf{y}=\{y_1,y_2,\ldots,y_k\}^T$. It has been proved [3] that the matched filter bank sacrifices no information relevant to demodulation. Hence y(t) can be replaced by $\mathbf{y}$ without any loss in system performance. Most MUDs therefore have the matched filter as the front end.

With the matched filter front end, the objective of MUD can be stated as follows:

Given the statistic $\{y_1,y_2,\ldots,y_k\}$ at the output of the matched filter, find an estimate for the transmitted bits $\{b_1,b_2,\ldots,b_k\}$ that minimizes the probability of error.

SECTION II

The Decorrelating Detector

While concluding Section 2.1 it was noted that the matched filter bank may decode erroneously even in the absence of background AWGN. This is not a very attractive property for any receiver. An optimal receiver must be capable of decoding the bits error-free when the noise power is zero. In this section the decorrelating detector is investigated. This detector makes use of the structure of MAI to improve the performance of the matched filter bank. The decorrelating detector falls into the category of linear multiuser detectors. This fact will be substantiated as this section progresses.

2.0 Receiver Operation

As shown in figure 2.1, the decorrelating detector operates by processing the output of the matched filter bank with the $\mathbf{R}^{-1}$ operator where $\mathbf{R}$ is the cross-correlation matrix as defined in (0.4).

The output of the decorrelating detector is given by

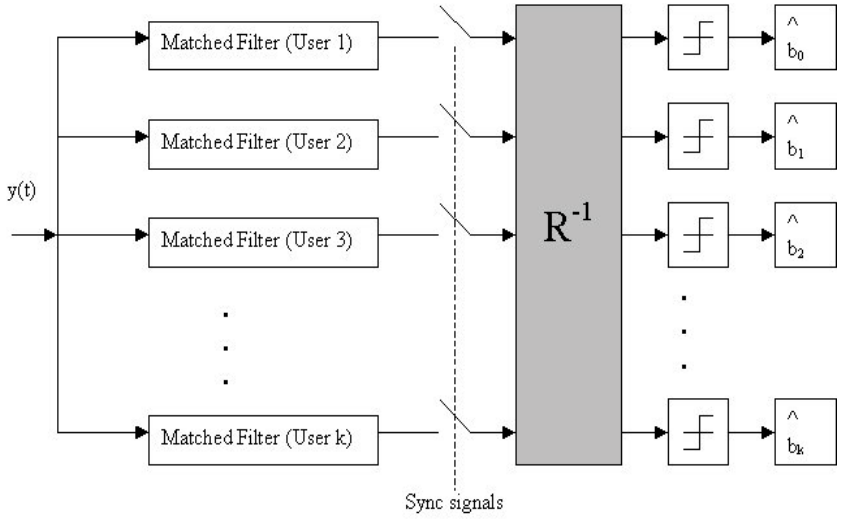

Figure 1.1 Decorrelating Detector

$$\hat{\mathbf{b}} = \operatorname{sgn}\left(\mathbf{R}^{-1}\left(\mathbf{RAb}+\mathbf{n}\right)\right) \quad \text{.....(2.1)}$$

$$= \operatorname{sgn}\left(\mathbf{Ab}+\mathbf{R}^{-1}\mathbf{n}\right) \text{.....(2.2)}$$

When the background noise is absent, i.e., $N_o=0$,

$$\hat{\mathbf{b}} = \operatorname{sgn}\left(\mathbf{Ab}\right) \text{.....(2.3)}$$

i.e., $$\hat{\mathbf{b}} = \mathbf{b} \text{.....(2.4)}$$

Hence, we observe that in the absence of background noise the decorrelating detector achieves perfect demodulation unlike the matched filter bank. One advantage of the decorrelating detector is that it does not require knowledge of the received signal amplitudes unlike the detector described in the next section.

Consider the two user case, with $\mathbf{R} = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$, where

ρ is the cross-correlation between the normalized signature waveforms of user 1 and user 2.

The decoded bits are given by (2.1),

$$\hat{\mathbf{b}} = \operatorname{sgn}\left(\mathbf{R}^{-1}\,\mathbf{y}\right) \text{.....(2.5)}$$

$$\hat{\mathbf{b}} = \operatorname{sgn}\left(\frac{1}{1-\rho^2}\begin{bmatrix} 1 & -\rho \\ -\rho & 1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \text{.....(2.6)}$$

$$\hat{\mathbf{b}} = \operatorname{sgn}\left(\begin{bmatrix} \frac{1}{1-\rho^2}y_1 & -\frac{\rho}{1-\rho^2}y_2 \\ -\frac{\rho}{1-\rho^2}y_1 & +\frac{1}{1-\rho^2}y_2 \end{bmatrix}\right) \text{.....(2.7)}$$

We see that the decorrelating receiver performs only linear operations on the received statistic $\mathbf{y}$ and hence it is indeed a linear detector. The decorrelating detector is proved to be optimal under 3 different criteria: least squares, near-far resistance and maximum-likelihood [3].

As in the previous section the bit error rate plots have been obtained for the 2 uses and 10 user cases and are shown in figure 2.2. The simulation scenario is described in section 0.3. Comparing figure 2.2 and figure1.2, we

see that for the 10 user case, as the SNR increases, the performance of the decorrelating detector is better. Again, in the simulations, perfect power control was assumed. Figure 3.4, figure 3.5 and figure 3.6 show a comparison of the performance of the matched filter bank and the decorrelating detector. It is observed that at low SNRs the matched filter performs better. Hence, the decorrelating detector is not an optimal (in terms of B.E.R) detector.

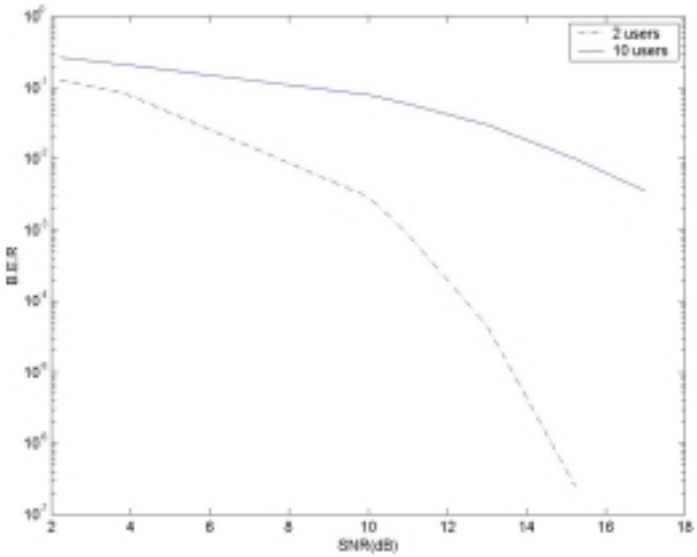

Figure 2.2 B.E.R performance of the decorrelating detector.

SECTION III

The MMSE Linear Detector

In section II we noted that the only information required by the decorrelating detector was the cross-correlation matrix **R** of the spreading sequences. At low SNRs, the matched filter bank performs better than the decorrelating detector as observed from figure 3.6. Hence, it might be possible to improve the performance by incorporating some SNR information in the MUD algorithms. In this section, one such approach is investigated where the mean squared error between the output and data is minimized. The detector resulting from the MMSE (minimum mean square error) criterion is a linear detector.

Two different adaptive approaches of the MMSE linear detector are also studied at the end of this section. One of the approaches requires no prior information of the SNRs or the signature waveforms but requires a training sequence to adapt and compute the optimum weights to be applied on the received statistic. The other approach does not need a training sequence but requires exact knowledge of the signature sequence.

3.0 Receiver Operation

Being a linear detector like the decorrelating detector, the MMSE receiver also weights the received statistic **y** with a weight vector **w** to form the decision statistic. The receiver structure for user m is shown in figure 3.1.

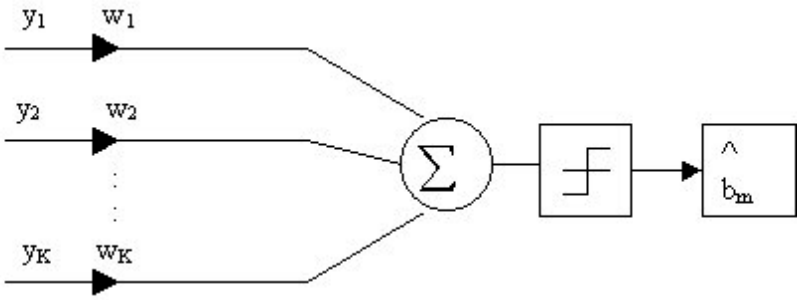

Figure 3.1 MMSE linear transformation for user m

It has been proved that minimizing the MSE at the output of the linear transformation is equivalent to maximizing the SIR at the output of the linear transformation [3]. The optimal value of the [w1,w2,…wk] that minimizes the MSE between the weighted received statistic and the transmitted bit is derived in the next section.

3.0.1 Optimal Weights for an MMSE Linear Detector in an AWGN Channel

The MMSE linear detector for user 1 determines a waveform c1(t) such that the MSE error between the transmitted bit and the correlation between c1(t) and the received signal y(t) is minimized. The objective function (the mean square error in this case) is defined as

$$\Psi(c1) = E\left\{\left(b_1 - \langle c1, y\rangle\right)^2\right\} \quad \text{.....(3.1)}$$

In the finite dimensional representation (3.1) can be expressed as

$$\Psi(w_1, w_2, ..., w_K) = E\left\{\left(b_1 - \sum_{i=1}^{K} w_i y_i\right)^2\right\} \quad \text{.....(3.2)}$$

Where $\{w_1, w_{2,\dots,} w_K\}$ are the weights operating on the received statistic $\{ y_1, y_{2,\dots,} y_K\}$.

Representing (3.2) in a compact and convenient matrix notation,

$$\Psi(\mathbf{w}) = E\left\{\left(b_1 - \mathbf{w}^T\mathbf{y}\right)^2\right\} \quad \text{.....(3.3)}$$

Using linearity of the Expectation operator,

$$\Psi(\mathbf{w}) = E\left(b_1^2\right) - E\left(2b_1\mathbf{w}^T\mathbf{y}\right) + E\left\{\left(\mathbf{w}^T\mathbf{y}\right)\left(\mathbf{w}^T\mathbf{y}\right)^T\right\} \quad \text{.....(3.4)}$$

Since the bits of user 1 are i.i.d, $E\left(b_1^2\right) = 1$. Therefore,

$$\Psi(\mathbf{w}) = 1 - 2\mathbf{w}^T E\left(b_1\mathbf{y}\right) + E\left\{\mathbf{w}^T\mathbf{y}\mathbf{y}^T\mathbf{w}\right\} \quad \text{.....(3.5)}$$

$$= 1 - 2\mathbf{w}^T E\left(b_1\mathbf{y}\right) + \mathbf{w}^T E\left\{\mathbf{y}\mathbf{y}^T\right\}\mathbf{w} \quad \text{.....(3.6)}$$

From (2.10), we have

$$\mathbf{y} = \mathbf{RAb} + \mathbf{n} \quad \text{.....(3.7)}$$

Consider, $E(b_1\mathbf{y}) = E(b_1\mathbf{RAb} + b_1\mathbf{n})$ .....(3.8)

$$= \mathbf{RA}E\left(b_1\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix}\right) + E(b_1\mathbf{n}) \quad \text{.....(3.9)}$$

$$= \mathbf{RA}\begin{bmatrix} E(b_1^2) \\ E(b_1b_2) \\ \vdots \\ E(b_1b_K) \end{bmatrix} + b_1E(\mathbf{n}) \quad \text{.....(3.10)}$$

Since the bits of user 1 are i.i.d and are uncorrelated with the bits of other users we have,

$$E(b_1b_K) = \begin{cases} 0 & ,i \neq j \\ 1 & ,i = j \end{cases} \quad \text{.....(3.11)}$$

Using (3.11) and the fact that the noise $\mathbf{n}$ is zero mean i.e.,$E(\mathbf{n})=0$ in (3.10),

$$E(b_1\mathbf{y}) = \mathbf{RA}[1\ 0\ 0\cdots 0]^T \quad \text{.....(3.12)}$$

Using the definition of $\mathbf{A}$ and $\mathbf{R}$ from (0.4)and (2.9),

$$E(b_1\mathbf{y}) = \begin{bmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1K} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{21K} \\ \vdots & \vdots & \cdots & \vdots \\ \rho_{K1} & \rho_{K2} & \cdots & \rho_{KK} \end{bmatrix}\begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & A_K \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{..(3.13)}$$

$$\therefore E(b_1\mathbf{y}) = \begin{bmatrix} \rho_{11}A_1 \\ \rho_{21}A_1 \\ \vdots \\ \rho_{K1}A_1 \end{bmatrix} \quad \text{.....(3.14)}$$

Now consider the second expectation term in (3.6),

$$E\{\mathbf{yy}^T\} = E\{(\mathbf{RAb})(\mathbf{RAb})^T\} + E(\mathbf{nn}^T) \quad \text{[using (3.7)].....(3.15)}$$

$$= E\{\mathbf{RAbb}^T\mathbf{A}^T\mathbf{R}^T\} + N_o\mathbf{R} \quad \text{[using (1.3)]......(3.16)}$$

Using the fact that $\mathbf{A}$ and $\mathbf{R}$ are symmetric matrices, we get

$$E\{\mathbf{yy}^T\} = \mathbf{RA}E\{\mathbf{bb}^T\}\mathbf{AR} + N_o\mathbf{R} \quad \text{.....(3.17)}$$

$$= \mathbf{RA}^2\mathbf{R} + N_o\mathbf{R} \quad \text{.....(3.18)}$$

Substituting (3.14) and (3.18) in (3.6),

$$\boxed{\Psi(\mathbf{w}) = 1 - 2\mathbf{w}^T[\rho_{11}A_1 \quad \rho_{21}A_1 \ \cdots \rho_{K1}A]^T + \mathbf{w}^T(\mathbf{RA}^2\mathbf{R} + N_o\mathbf{R})\mathbf{w}} \quad \text{..(3.19)}$$

Equation (3.19) gives the objective function (MSE) that should be minimized according to the MMSE criterion. Performing a matrix derivative operation on (3.19) we get,

$$\nabla_w\Psi(\mathbf{w}) = -2[\rho_{11}A_1 \quad \rho_{21}A_1 \ \cdots \rho_{K1}A]^T + 2(\mathbf{RA}^2\mathbf{R} + N_o\mathbf{R})\mathbf{w} \quad \text{.....(3.20)}$$

We used the fact that $(\mathbf{RA}^2\mathbf{R} + N_o\mathbf{R})$ is a symmetric matrix to get (3.20). The optimum weights that minimize the MSE can be obtained by equating $\nabla_w\Psi(\mathbf{w})$ to zero. Hence,

$$-2[\rho_{11}A_1 \quad \rho_{21}A_1 \ \cdots \rho_{K1}A]^T + 2(\mathbf{RA}^2\mathbf{R} + N_o\mathbf{R})\mathbf{w}_{opt} = 0 \quad \text{....(3.21)}$$

Solving (3.21) the optimal weights are obtained as

$$\mathbf{w}_{opt} = \left(\mathbf{R}\mathbf{A}^2\mathbf{R} + N_o\mathbf{R}\right)^{-1}\left[\rho_{11}A_1 \quad \rho_{21}A_1 \quad \cdots \rho_{K1}A_1\right]^T \qquad \text{.....(3.22)}$$

To calculate the optimal weights for user m, just replace ρ_{i1} by ρ_{im} for all i and replace A_1 by A_m in (3.22). (3.22) can be written in a more general and compact form as

$$\mathbf{w}_{opt} = \left(\mathbf{R} + N_o\mathbf{A}^{-2}\right)^{-1} \qquad \text{.....(3.23)}$$

where $N_o\mathbf{A}^{-2} = diag\left\{\frac{N_o}{A_1^2}, \frac{N_o}{A_2^2}, ..., \frac{N_o}{A_K^2}\right\}$ .....(3.24)

(3.23) explains the operation of the receiver. The receiver just weights the received statistic with $\mathbf{w}_{opt}$ and makes a decision as shown in figure 3.1. This leads to the receiver architecture shown in figure 3.2.

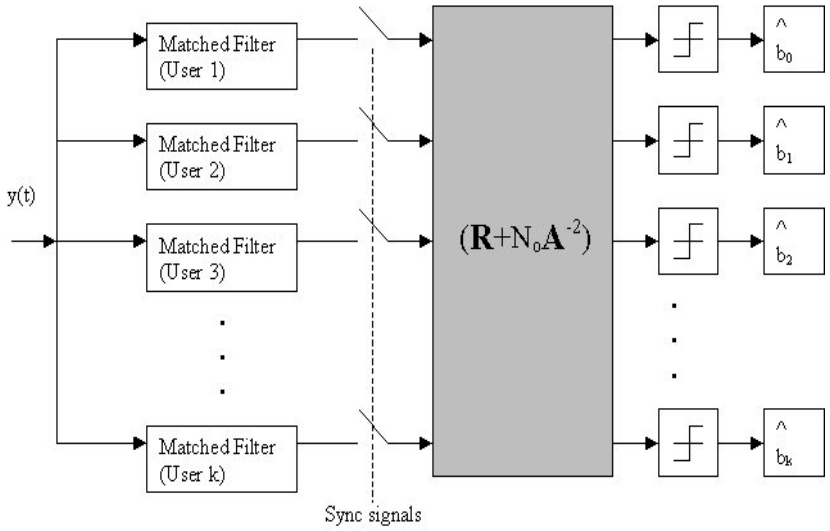

Figure 3.2 MMSE linear detector

It is shown in [3] that the MMSE receiver maximizes the SIR at the output of the transformation shown in the above figure. Figure 3.3 shows the performance of the MMSE linear detector in an AWGN channel. The simulation scenario is identical to the one used in the previous sections. For the sake of comparison the bit error rates of the detectors described so far have been plotted in figures 3.4, 3.5 and 3.6.

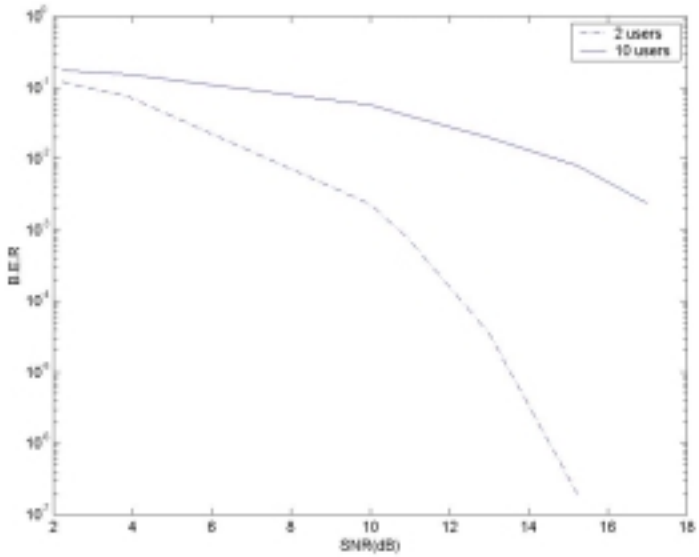

Figure 3.3 B.E.R performance of the MMSE linear detector

3.1 Adaptive Implementations

As seen in sections II and II, the implementations of the decorrelating detector and the MMSE linear detector involves matrix inversion operations. If the number of users become large the size of the matrix to be inverted grows and hence the computation time of such a matrix inversion becomes unacceptable. For the decorrelating detector R^{-1} can be pre-computed and stored. However for an asynchronous system, R is time varying and hence such pre-computation will not work. The MMSE

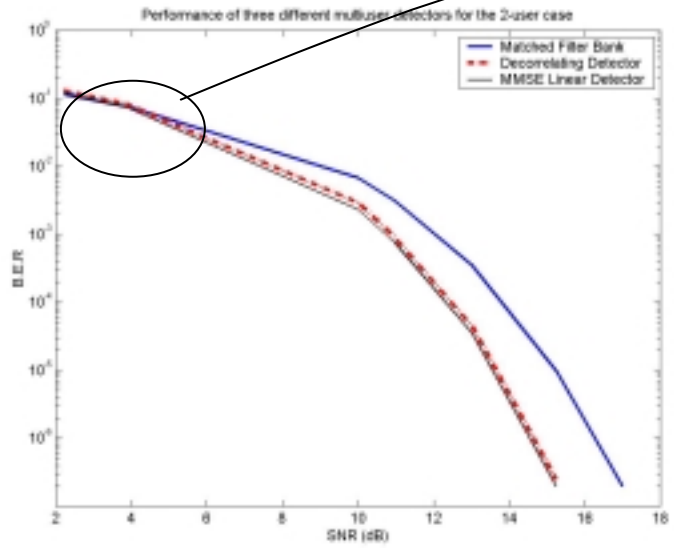

Figure 3.4.BER plots for the detectors described in section I ,II and III (2 user case)

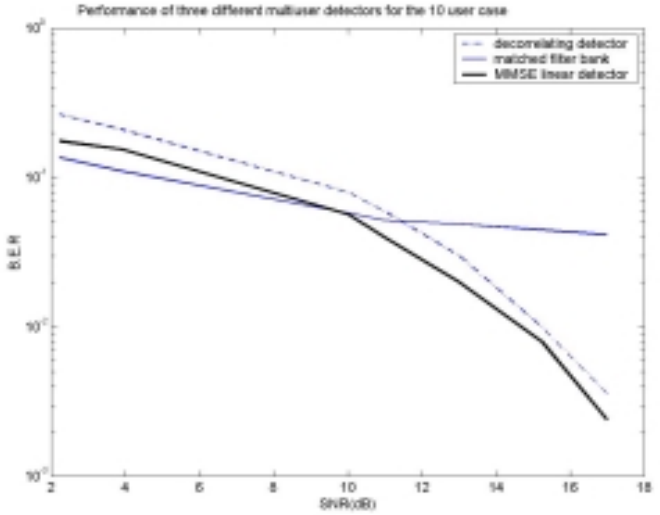

Figure 3.5 BER plots for the detectors described in section I ,II and III (10 user case)

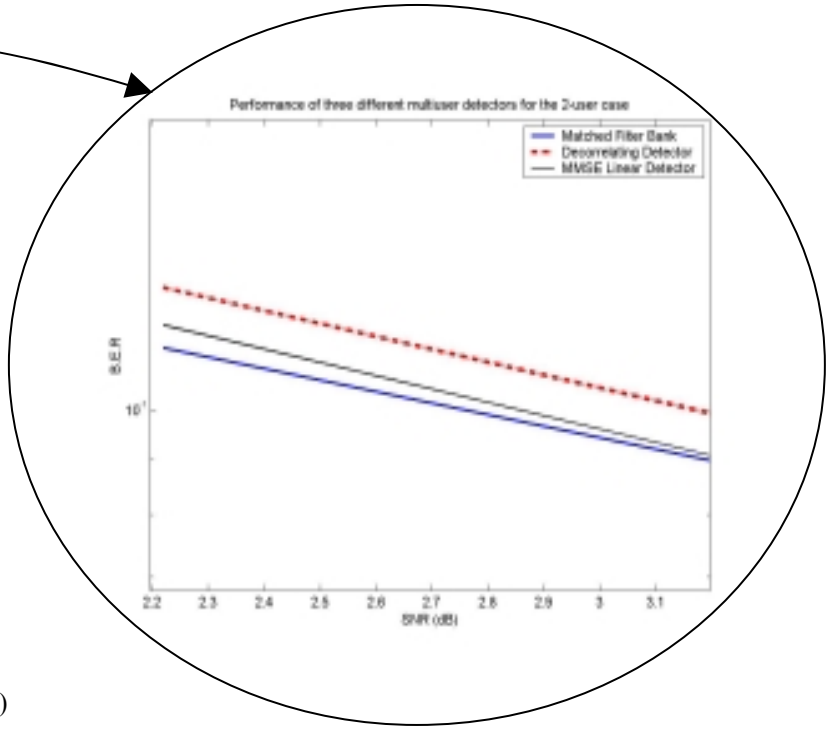

Fig 3.6 A zoomed version of the BER plot of the figure 3.4 showing the performance of the linear detectors at low SNRs

detector requires the SNR information and hence again precomputation of the matrix inverse is not a feasible solution. Also, getting good estimates of the SNR is not temporally efficient. Therefore, it would be nice if there was some way to eliminate the need to compute matrix inverses and the need to have apriori information (signature sequences) and other additional information (SNR) for decoding. This objective can be realized through adaptive MUD algorithms. Adaptive algorithms "learn" the desired filter response from the received signals. There are different approaches to implement the "learning" capability. Two approaches will be studied in the next sub-sections. The first approach does not any apriori knowledge but calls for a training sequence. The second approach does not require any training sequence but requires exact knowledge of the signature sequences of the users and also takes longer to converge.

3.1.1 Adaptive MMSE Linear Detection (LMS Algorithm)

Before the LMS (least mean square) algorithm is described, a brief review of the *gradient descent optimization* algorithm is presented. A more detailed presentation of this algorithm can be found in [3,4]

The gradient descent algorithm is used for the optimization of convex penalty functions. Consider the optimization of the following convex penalty function :

$$\Psi(u) = E[g(X,u)] \quad \text{.....(3.25)}$$

where E is the expectation operator , X is a random variable and u is the parameter to be optimized (X and u could be vectors). If ψ is convex, then according to the gradient descent algorithm, it is possible to converge to the minimum valuse of ψ by starting at any point u_o and following the direction opposite to the gradient $\nabla\psi$ (steepest descent). The update rule is then given by

$$u_{j+1} = u_j - \mu\nabla\Psi(u_j), \text{ where } \mu \text{ is the step size.....(3.26)}$$

However, if the distribution of X is not known then neither the penalty function given by (3.25) nor its gradient can be computed. But if a number of independent observations of X are available then it would be possible to get an estimate of the distribution of X and calculate the gradient of the penalty function and use the update rule given in (3.26). Therefore, at each iteration we could replace the gradient of the penalty function, $\nabla\psi=E[\nabla g(X,u)]$ by its approximate value $\nabla g(X_{j+1},u)$. This is called the stochastic gradient descent algorithm. The update rule for the stochastic gradient is thus modified as

$$u_{j+1} = u_j - \mu\nabla g(X_{j+1}, u_j) \quad \text{.....(3.27)}$$

If the step size is infinitesimally small, then the deviations on either side of the mean tend to cancel out and the trajectory of the stochastic descent will almost follow the steepest descent trajectory. For the special case of quadratic cost functions, the stochastic descent algorithim is also known as the least mean square (LMS) algorithm.

For the case of MMSE multiuser detection in CDMA systems, the convex penalty function is given by (3.3) . Hence

$$g(\mathbf{X},\mathbf{w}) = \left(b_1 - \mathbf{w}^T\mathbf{y}\right)^2 \text{ , where } \mathbf{X}=(b_1, \mathbf{w}) \quad \text{.....(3.28)}$$

Differentiating w.r.t $\mathbf{w}$, $\nabla g(\mathbf{X},\mathbf{w}) = -2\left(b_1 - \mathbf{w}^T\mathbf{y}\right)\mathbf{y}$ .....(3.29)

Since $X_j=(b_1[j], y[j])$, the update rule in (3.27) becomes

$$\boxed{\mathbf{w}[j] = \mathbf{w}[j-1] - \mu\left(\mathbf{w}^T[j-1]\mathbf{y}[j] - b_1[j]\right)\mathbf{y}[j]} \quad \text{.....(3.30)}$$

It is seen that we need to know the data bits b1 in order to implement the LMS algorithm. This requirement is handled by sending a training sequence at the beginning of each transmission. Once the training sequence has been sent, the adaptive algorithm can be allowed to run with the decisions made by the detector instead of the true transmitted data. This mode of operation is called *decision directed* operation.This might fine tune the weights if the SNR is high enough. However if the SNR is very low, the decisions of the detector are not reliable enough and may cause the weight to change drastically from the optimal value. In the simulation results presented in this report, the decision directed mode was not used. Once the training bits are sent, the weights were not changed.

The longer the training sequence, the closer are the computed weights to the optimal value given by (3.23). The training bits however are a overhead and the number of training bits needs to be as small as possible in order to maintain system efficiency. Hence there is a tradeoff between efficiency and error performance that needs to be considered when deteremining the number of training bits that needs to be used in a system. This is observed by comparing figure 3.7 and figure 3.9 which show that the weights w1 and w2 (in the 2 user case) do not converge to the optimal value with just 1000 training bits. However when 5000 training bits are sent, the weight vectors were closer to the optimal value (as observed from the MSE curve, figures 3.8 and 3.10). The simulations were run with an SNR of 10dB and the optimal values required to calculate the MSE were obtained from (3.23).

Apart from the number of training bits another important parameter that affects the performance and convergence speed of the the LMS algorithm is the step size. A larger step size makes the algorithm converge faster but has a higher ripple around the optimal value. This can be observed by comparing figures 3.9and 3.13 and also figures 3.10 and 3.14.

Performance of the LMS linear detector with a step size μ=0.001

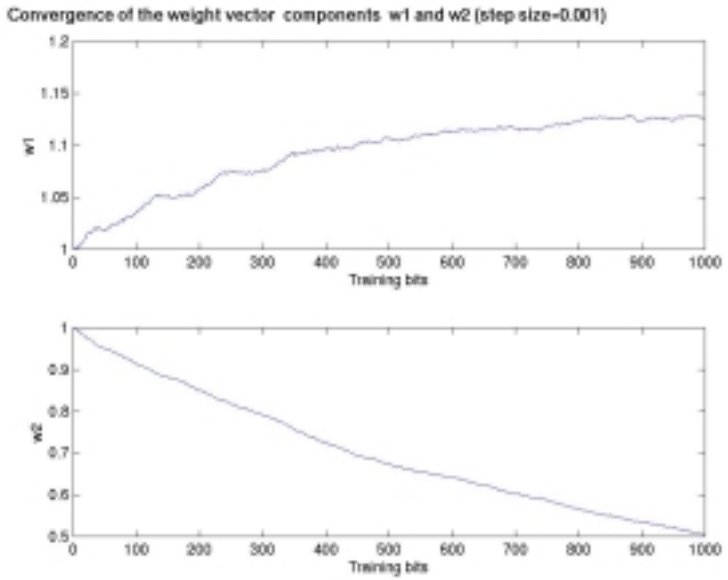

Figure 3.7 Weight vector convergence for 1000 training bits

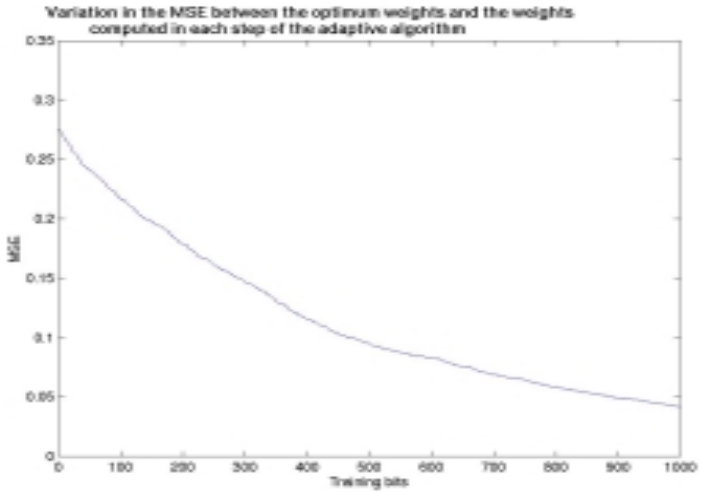

Figure 3.8 MSE variation for 1000 training bits

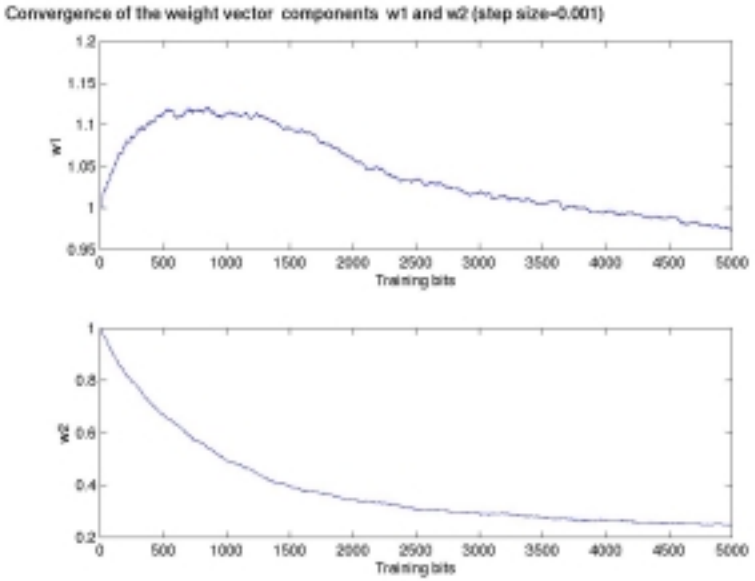

Figure 3.9 Weight vector convergence for 5000 training bits

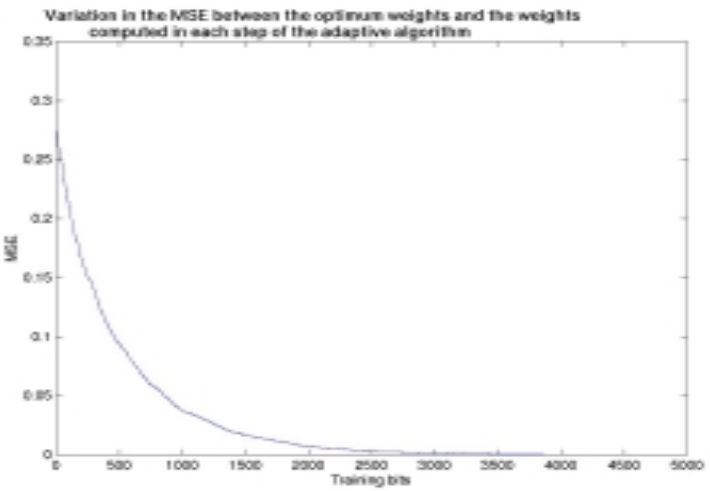

Figure 3.10 MSE variation for 5000 training bits

Performance of the LMS linear detector with a step size μ=0.01

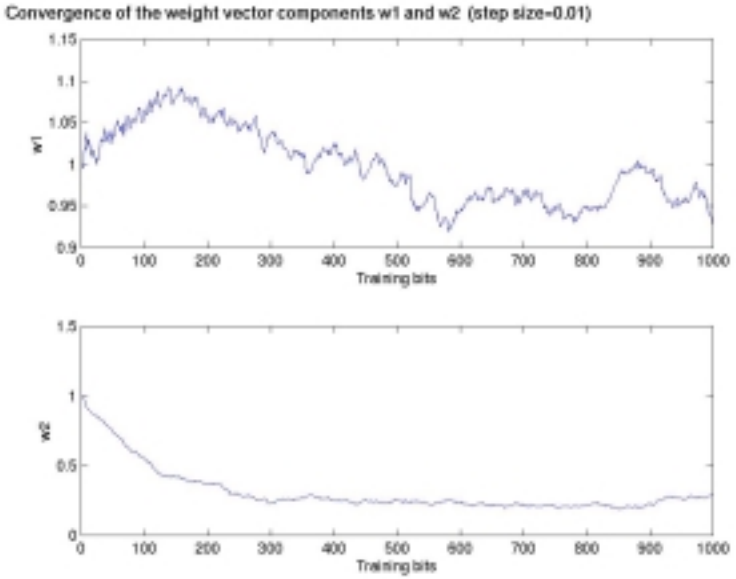

Figure 3.11 Weight vector convergence for 1000 training bits

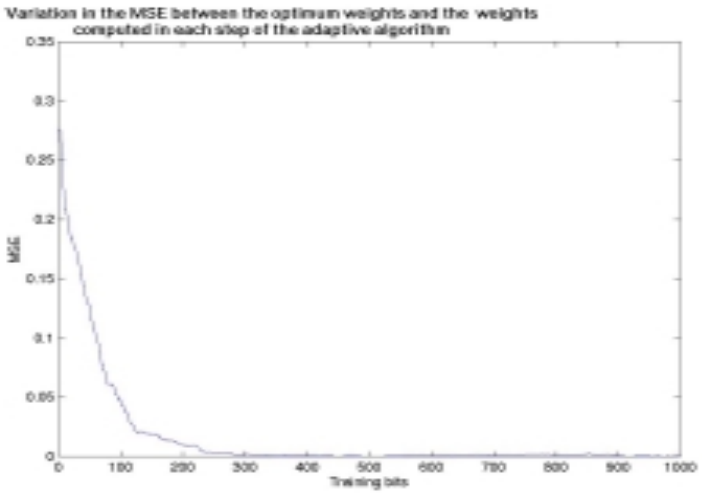

Figure 3.12 : MSE variation for 1000 training bits

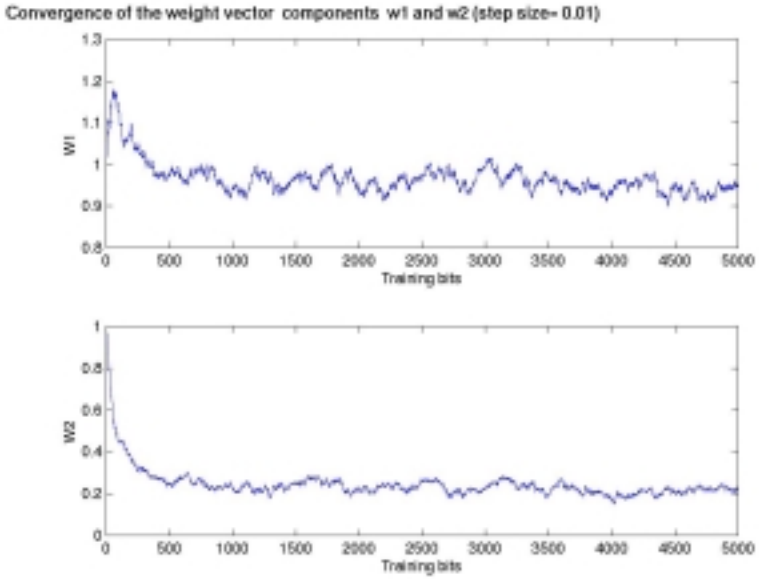

Figure 3.13 Weight vector convergence for 5000 training bits

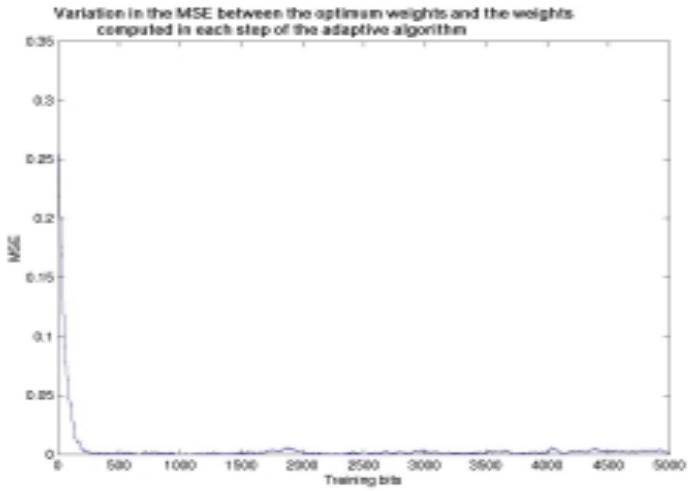

Figure 3.14 : MSE variation for 5000 training bits

Observing figure 3.7 and figure 3.11 we see that with a larger step size, the weights converge to the optimal value with a smaller number of training bits but at the cost of a higher residual error (weights do not converge to optimal value with a step size of 0.001 and 1000 training bits but if the step size is increased to 0.01, the weight vectors converge around the optimal value with just 1000 bits). Conversely, a smaller step size takes longer to converge(requires more training bits) .

From the above discussion, it is clear that it would be nice to progressively decrease the step size as the LMS algorithm proceeds. A high value of step size should be used initially to cause fast convergence of the the algorithm and then in the latter iterations a smaller step size should be used to minimize the ripple around the optimal value (residual error). But great care should be excercised in using adaptive step sizes. Sometimes the step size may become really small as the algorithm progresses and the weights may never converge to the optimal value. One method of progressively shrinking the step size is to multiply a fixed step size by γ^i where i is the iteration number and γ is a number just smaller than 1. Hence the update rule is given by

$$\mathbf{w}[j] = \mathbf{w}[j-1] - \gamma^i \mu \left(\mathbf{w}^T[j-1]\mathbf{y}[j] - b_1[j]\right)\mathbf{y}[j] \quad \ldots\ldots\ldots\ldots (3.31)$$

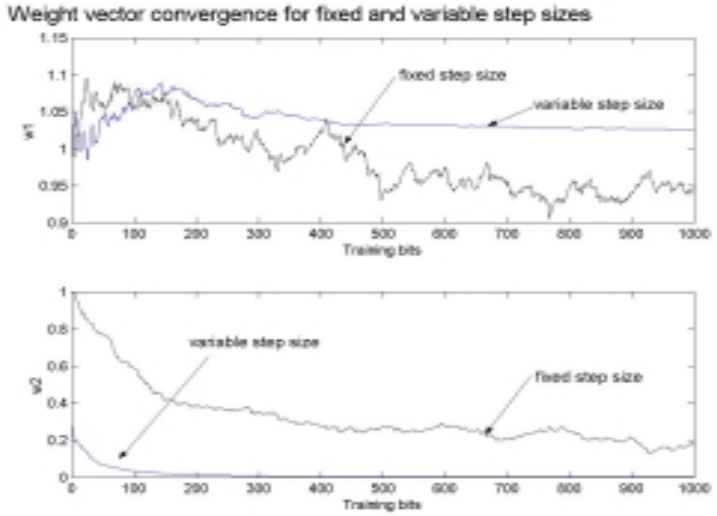

Figure 3.15 Weight vector convergence for the update rule in (3.31)

The convergence of the weights with this adaptive step size is shown in figure 3.15. In the simulations, γ =0.995 and μ=0.01. We can see that the step size becomes very small as the algorithm progresses and the weights do not converge. Hence, this is not a suitable strategy for varying the step size in this case.

Another method of progressively decreasing the step size is to have a step size of the form 1/i, where i is the iteration number. Hence the update rule is modified as

$$\mathbf{w}[j] = \mathbf{w}[j-1] - \frac{1}{i}\left(\mathbf{w}^T[j-1]\mathbf{y}[j] - b_1[j]\right)\mathbf{y}[j] \quad \ldots\ldots\ldots\ldots (3.32)$$

The convergence plots for this step size is given in Figure (3.16). Also plotted are convergence plots with a fixed step size of 0.01 for comparison. We see that this method of varying the step size converges very fast to the vicinit of the optimal value. For the a step size of the form 1/i it has been proved [3] that the update rule given in (3.32) will always converge to the optimum value.

If the channel changes often, then training bits need to be sent periodically. The choice of step size, whether it should be fixed or stationary, depends on the application, channel and other cost factors.

Since, the detector requires no knowledge of the signature sequence, why

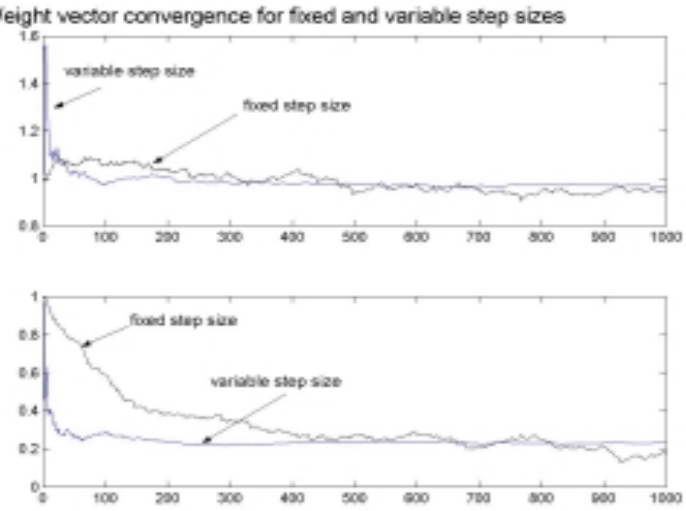

Figure 3.16 Weight vector convergence for the update rule in (3.32)

doesn't the detector converge to some other users MMSE detector (say the user with the strongest power)? This is because of the training sequence which governs the desired users LMS algorithm. Hence, each user should have a different training sequence. Hence, we see that the LMS MUD is not limited by the near far problem.

3.1.2 Blind Adaptive Multiuser Detection

The convergence of the LMS algorithm rests on the fact that the received amplitudes, cross-correlations and channel conditions remain constant. If any of these parameters change suddenly (a strong interfere is powered on or the channel suddenly goes into a deep fade), then in the decision directed mode of operation the weights may not converge because the detector starts making faulty decisions or in the fixed mode of operation the weights are no longer optimal. This calls for the training sequence to be sent again. Thus, the detector needs to be able to detect sudden changes in the system environment and request for the training sequence to be sent again. This is a cumbersome procedure requiring a lot of overhead. Hence, it is desirable to have an adaptive algorithm that operates without having to know the data. Algorithms that operate without knowledge of the channel input are known as *blind* algorithms (the algorithm is blind to the data). Blind adaptive multiuser detection was first introduced by Honig et al in 1995 [5]. Before the blind MUD algorithm is presented the *canonical* representation of linear multiuser detection introduced in [5] is discussed.

The blind MUD approach is based on decomposing the linear MUD filter response into two orthogonal components. One of the components is equal to the signature waveform of the desired user. Consider the linear detector of user 1 which is characterized by the filter response c_1,

$$\hat{b}_1 = \mathrm{sgn}(< y, c_1 >) \quad \text{.....(3.33)}$$

The canonical representation of c_1 is

$$c_1 = s_1 + x_1 \quad \text{.....(3.34)}$$

where $<s_1, x_1>=0$ .....(3.35)

s_1 in (3.34) and (3.35) is the signature waveform of user 1. Therefore, according to the canonical representation every linear MUD is characterized by its orthogonal signal x_1 (since s_1 is assumed to be known). Given a linear transformation d_1 the orthogonal component is given by

$$x_1 = \frac{1}{< s_1, d_1 >} d_1 - s_1 \quad \text{.....(3.36)}$$

This can be verified by applying (3.35) to (3.36). Using (3.34) in (3.33) we have,

$$\hat{b}_1 = \mathrm{sgn}(< y, s_1 + x_1 >) \quad \text{.....(3.37)}$$

Using the definition of y from (0.1)

$$\hat{b}_1 = \mathrm{sgn}(A_1 b_1 + \sum_{k=2}^{K} A_k b_k (\rho_{1k} + < s_k, x_1 >) + \tilde{n}_1 \quad \text{.....(3.38)}$$

Just as done in section 3.0.1, it can be proved that

$$E\left[\tilde{n}_1^{\,2} \right] = 1 + \|x\|^2 \quad \text{.....(3.39)}$$

The variance at the output of the linear transformation in (3.37) is given as

$$E\{(< y, s_1 + x_1 >)^2\} \quad \text{.....(3.40)}$$

This is also referred to as the *output energy* in literature. Using the definition of $<y,s_1+x_1>$ in (3.40) we can express the output energy as

$$E\{(< y, s_1 + x_1 >)^2\} = A_1^2 + \sum_{k=2}^{K} A_k^2 (\rho_{1k} + < s_k, x_1 >)^2 + N_o \{1 + \|x\|^2\}$$

.....(3.41)

We have used the fact that the bits of user 1 are uncorrelated with the bits of the other users and the noise. The first term in (3.41) is the signal energy and the two other terms represent the interference energy (MAI + noise). Due to the canonical representation we can see that the signal energy is transparent to the choice of x_1. We can intuitively see that choosing x_1 to minimize the output energy would be a sensible idea since this would just minimize the interference energy. This type of detector is referred to as the *minimum output energy* (MOE) detector. Denote the minimum output energy as

$$\mathrm{MOE}(x_1) = E\{(< y, s_1 + x_1 >)^2\} \quad \text{.....(3.42)}$$

From section 3.01, the minimum MSE is given by

$$\mathrm{MSE}(x_1) = E\{(A_1 b_1 - < y, s_1 + x_1 >)^2\} \quad \text{.....(3.43)}$$

i.e., $MSE(x_1) = \sum_{k=2}^{K} A_k^2 \left(\rho_{1k} + < s_k, x_1 >\right)^2 + N_o\left(1 + \|x\|^2\right)$....(3.44)

=> $MSE(x_1) = MOE(x_1) - A_1^2$ (using (3.41)) ……(3.45)

∴ We can see that the MSE and MOE differ only by a constant (thanks to the canonical representation). Hence, the arguments that minimize both the functions are the same. This is a crucial observation from the point of view of the adaptive implementation. It tells us that if we try to minimize the MOE (same as minimizing the MSE), we do not need any training sequence to implement the gradient descent algorithm (since the MOE does not depend on the data). Hence we end up with an adaptive MMSE detector without the need for any training sequence. This is the basis for the blind adaptive multiuser detector.

The steepest descent algorithm (described in the previous subsection) is again used to derive the update rule for the adaptive algorithm. We just need to update the orthogonal component to s_1 since we are trying to obtain the minimum of the MOE by changing just x_1. Therefore we just need to follow the steepest descent in the direction orthogonal to s_1. To do this we just take the gradient of the MOE and project it on the subspace orthogonal to s_1. The gradient of the MOE given in (3.42) is

$$\nabla MOE(x_1) = 2 < y, s_1 + x_1 > y \text{ ……..……..(3.46)}$$

The component orthogonal to s_1 is $y - <y,s_1> s_1$. Using this in (3.46) we get the gradient projected onto a subspace orthogonal to s_1 as

$$2< y,s_1+x_1>[y- <y,s_1> s_1]\text{……..……(3.47)}$$

The adaptive algorithm update is done once every T seconds, where T is the bit period. The received waveform is slotted into waveforms of duration T,

$$...y[i-1], y[i], y[i+1]\ldots$$

Define, $Z_{MF}[i] \overset{\Delta}{=} < y[i], s_1 > \text{……..……..……..(3.48)}$

$$Z[i] \overset{\Delta}{=} < y[i], s_1 + x_1[i-1] > \text{……(3.49)}$$

Using (3.49) and (3.48) in (3.47) and using the update rule in (3.27), the update rule for the blind adaptive MUD is given by

$$x_1[i] = x_1[i-1] - \mu\, Z[i]\left(y[i] - Z_{MF}[i] s_1\right) \text{……(3.50)}$$

Finite precision implementation of the above update rule can have effects that are not immediately visible but that have a cumulative effect. It may happen that finite precision round off errors drive the updates of x_1 to be outside the orthogonal space. This is shown in figure 3.17 for two different values of step sizes. The simulations were run with 2 users and perfect power control. On the y axis, a measure of the orthogonality between $x_1$and s_1 ($<x_1, s_1>$) is plotted for each iteration. Though $<x_1, s_1>$ is small, we can see that the value keeps increasing implying that x_1 keeps falling more and more outside the orthogonal subspace.

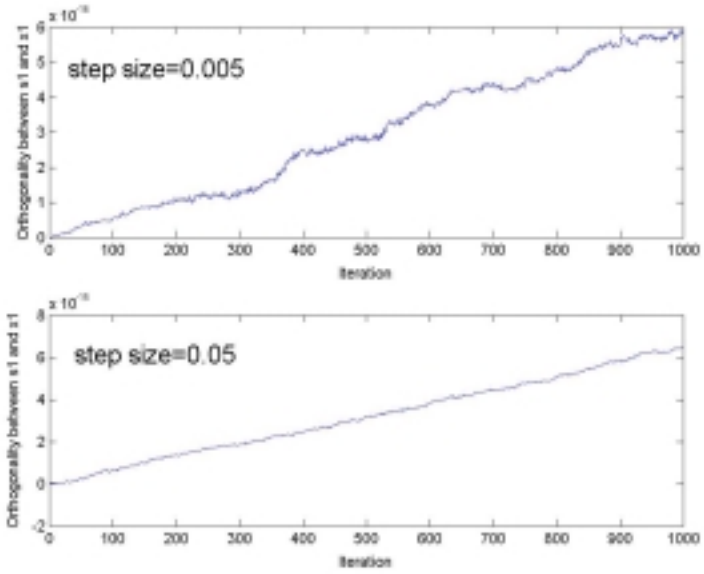

Figure 3.17 Finite precision implementation errors in the blind update rule

This problem can be solved by replacing the update x_1 by its orthogonal projection:

$$x_1[i] - <x_1[i], s_1[i]> s_1 \text{.....(3.51)}$$

Figure 3.18 plots the orthogonality between $x_1$and s_1 after replacing x_1 by its orthogonal projection. We now observe that the value of $<x_1, s_1>$ oscillates around the same value.

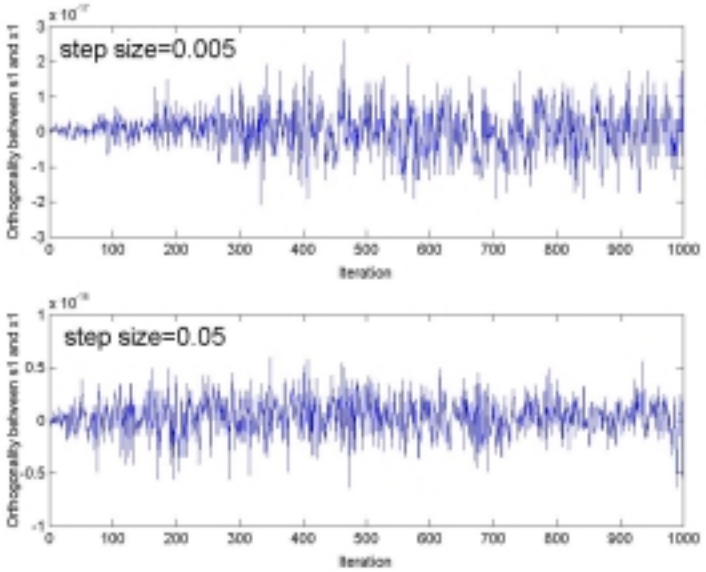

Figure 3.18 Effect of the substitution in (3.51) on finite precision errors

It was observed earlier that the first term in (3.41) represented the signal energy and the second term in (3.42) represents the interference energy. Therefore, the signal to interference ratio for user 1 can be written as

$$SIR = \frac{A_1^2}{\sum_{k=2}^{K} A_k^2 \left(\rho_{1k} + < s_k, x_1 >\right)^2 + N_o\left(1 + \|x\|^2\right)} \quad \ldots\ldots(3.52)$$

It can be inferred from looking at (3.41) that minimizing the output energy is the same as minimizing the denominator of (3.52). Therefore just like the MMSE detector the MOE detector also maximizes the output SIR (another way to prove that the two detectors are equivalent). Figure 3.19 shows how the SIR increases with each iteration (the SNR was 10dB with 10 users in the system).

Figure 3.20 shows the convergence of one of the components of x_1. By looking at the convergence plots of the LMS detector we observe that the blind detector takes a lot longer converge.

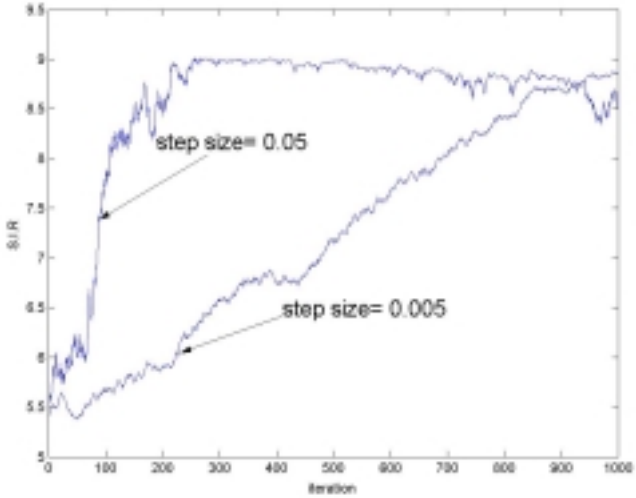

Figure 3.19 SIR at each iteration of the blind MUD algorithm

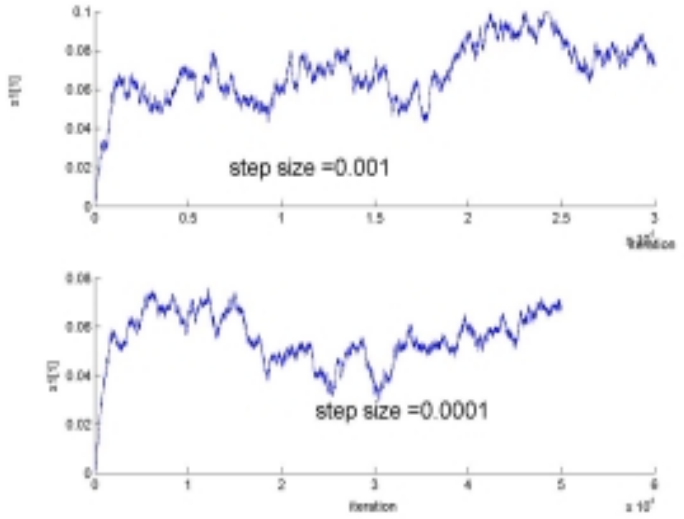

Figure 3.20 Convergence of 1 componant of x_1

It is not really fair to compare the LMS and blind detectors in terms of the bit error rate because the performance depends on how long the algorithm is allowed to converge(also the two algorithms do not behave the same way for the same step size). Once the algorithms converge, the error performance should ideally be that of the MMSE linear detector. But to give a rough idea, the BER error rates for the 2 user case are plotted in Figure 3.21. The LMS algorithm was given 1000 training bits and had a step size of 0.01. The blind detector had a step size of 0.005. Hence, the comparison is not really a good one, however it seen that the performance is almost the same.

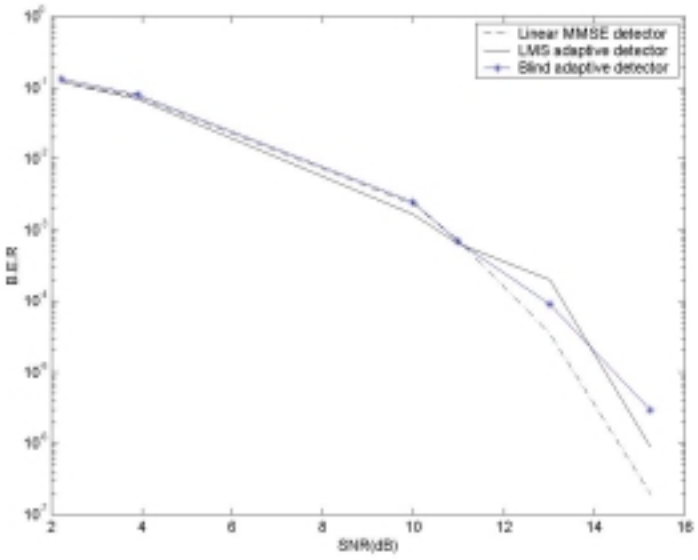

Figure 3.21 BER performance of the MMSE,LMS and blind detectors

CONCLUDING REMARKS

This report is a compilation of different approaches to linear multiuser detection. The requirement of this technology was motivated by studying the conventional detector. The matched filter bank just ignores the correlative structure of the MAI present in CDMA systems. Further, it was also shown that in the absence of noise, the conventional detector is a totally unreliable detector. This called for the need for better detectors.

The decorrelating detector was then introduced which takes the conventional detector one step further by incorporating the correlative structure of the MAI in the detection. However, it was noted that at low SNRs the conventional detector performed better than the decorrelating detector. This implied that the decorrelating detector could be improved upon.

The MMSE linear detector was then shown to take the decorrelating detector one step further by incorporating some SNR information along with the correlative structure of MAI. Thus, the performance was better than the decorrelating detector at low SNRs. It must also be noted that when the background noise is totally absent (infinite SNR), the MMSE operator $\left(\mathbf{R} + \mathrm{N}_o \mathbf{A}^{-2}\right)^{-1}$ reduces to $\left(\mathbf{R}\right)^{-1}$, which the decorrelating operator. Hence the decorrelating detector can be thought of as an asymptotic instance of the MMSE linear detector. However, the implementation of the MMSE linear detector required knowledge of received signal amplitudes (to calculate the SNR) apart from the correlation matrix and timing information (required by the decorrelating detector).

Two different adaptive approaches to linear MMSE detection that operate without knowledge of received amplitudes was then discussed. The LMS detector required no knowledge of the signature sequences but called for training sequences. The blind detector required only the same knowledge as required by the conventional detector, the signature sequences and the timing. The convergence properties of the two adaptive algorithms were also studied. When compared to the LMS algorithm the blind algorithm takes a very long time to converge.

The choice of the MUD algorithm depends on a lot of factors like the application, channel information available, availability of training sequences, complexity cost and overhead involved. To pick one out which is the optimal one or the best one is not an easy task!

APPENDIX A : The 10 gold codes used in the simulations

User	Code
1	1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0
2	1 1 1 1 1 0 1 1 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 0 0
3	0 1 1 0 0 0 0 1 0 1 1 0 1 0 0 1 1 1 0 0 1 1 1 1 0 0 1 1 0 0 1
4	0 1 1 1 1 0 0 0 0 1 0 1 0 1 1 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1
5	0 1 0 0 1 0 1 0 0 0 1 0 1 0 1 0 1 1 1 0 1 0 0 0 1 1 1 0 1 1 1
6	0 0 1 0 1 1 1 0 1 1 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0 0 1 1 1 1 1
7	1 1 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1
8	0 1 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 1 1 1 0 0 1 1 0 1 1 1 0
9	0 1 0 1 0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1
10	0 0 0 1 1 1 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 1

The codes must be converted to 1s and -1s and then normalized to have unit energy before they can be used in the simulations.

APPENDIX B: The MATLAB codes used in the simulations.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%code for generating the m-sequence of length 31%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
%%%%%% enter tap weights as an array h=[h1,h2,..h6]
%%%%%%%%%%%
%h=[0 0 0 0 1 1];% 103 ---this is the primitive polynomial <103>
%h=[1 0 0 1 1 1];%147
h=[1 1 1 0 1];%75
%h=[0 0 1 0 1];%45
%h=[1 1 0 0 1 1];%163
%h=[1 0 0 0 0 1];%141
%%%%%% enter intial state of the register as an array u=[u1 u2 u3 u4 u5
u6]  %%%%%%%%%%%
%fid=fopen('/home/users/arun/sscdma/project/mseq31_primploy45.txt','a');
for i=1:31
  u=dec2bin(i,5);
  u=u-48;
   u(6)=0;
    output=[];

       for shift=1:31
   output=[output,u(1)];
         temp=u(1);
   for n=2:6
     u(n-1)=xor( u(n),(h(n-1)*temp) );
          end
  %u(1:6)
  end
  %output
  t=[output,output];
  if t(1,2:31)==t(1,2*(2:31)-1)
    charphase=output;
  end
end
```

```
save mseq31primpoly75.mat charphase;
u=charphase;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%code for generating the gold codes%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
N=31%pn code length
num_users=10;

load
'/home/users/arun/sscdma/project/mseq31primpoly45.mat'
u(1,:)=charphase;
load
'/home/users/arun/sscdma/project/mseq31primpoly75.mat'
u(2,:)=charphase;
fid=fopen('/home/users/arun/sscdma/project/codes.txt','a
+');
%shift and add
for j=1:num_users-2
   u(j+2,:)=xor(u(1,:),u(2,[j+1:end 1:j]));
end
for i=1:num_users
   for j=1:N
      fprintf(fid,'%d ',u(i,j));
   end
 fprintf(fid,'\n');
end
fclose(fid);

%convert to ones and minus 1's
u=2*u-ones(num_users,N);
u=u/sqrt(31);%normalize the auto-correlation

%calculate the cross-correlation matrix
for i=1:10
   for j=1:10
      R(i,j)=sum(u(i,:).*u(j,:));
   end
end
save  '/home/users/arun/sscdma/project/10goldcodes.mat'…
   u R;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%code for generating the matched filter (2 users) %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
N=31;
K=2;%%%%no of users
No=0.6;
Nodb=10*log10(1/No)
%No=N*No;
no_of_bits=1000;
%%%%get signature sequences
load '/home/users/arun/sscdma/project/mseq31primpoly45.mat'
a1=charphase;
load '/home/users/arun/sscdma/project/mseq31primpoly75.mat'
a2=charphase;
%%%%%%%generate the anitpodal sequence
a1=2*a1-1;
a2=2*a2-1;
%%%%%%%normalize energy of signature waveforms
a1=a1/sqrt(sum(a1.*a1));
a2=a2/sqrt(sum(a2.*a2));

A=[1 0;0 1];
rho=sum(a1.*a2);
R=[1 rho;rho,1];

bits=round(rand(K,no_of_bits));
b=2*bits-1;
n=sqrt(No)*randn(K,no_of_bits);

y=sign(R*A*b + n);
%convert to ones and
b_hat=(y+ones(K,no_of_bits))/2;

ber1=sum(xor(bits(1,:),b_hat(1,:)))/no_of_bits
ber2=sum(xor(bits(2,:),b_hat(2,:)))/no_of_bits
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%code for the matched filter bank(2 users)%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
N=31;
K=10;%%%%no of users
No=0.02;
Nodb=10*log10(1/No)
%No=N*No;
no_of_bits=100000;
%%%%get signature sequences
%load '/home/users/arun/sscdma/project/mseq31primpoly45.mat'
%a1=charphase;
%load '/home/users/arun/sscdma/project/mseq31primpoly75.mat'
%a2=charphase;
%%%%%%generate the anitpodal sequence
%a1=2*a1-1;
%a2=2*a2-1;
%%%%%%normalize energy of signature waveforms
%a1=a1/sqrt(sum(a1.*a1));
%a2=a2/sqrt(sum(a2.*a2));

A=eye(K);
load '/home/users/arun/sscdma/project/10goldcodes.mat'

bits=round(rand(K,no_of_bits));
b=2*bits-1;
n=sqrt(No)*randn(K,no_of_bits);

y=sign(R*A*b + n);
%convert to ones and
b_hat=(y+ones(K,no_of_bits))/2;

ber1=sum(xor(bits(1,:),b_hat(1,:)))/no_of_bits
ber2=sum(xor(bits(2,:),b_hat(2,:)))/no_of_bits
avg_ber=0.5*(ber1+ber2)
ber=sum(sum(xor(bits(1:K,:),b_hat(1:K,:)))/no_of_bits)/K
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%code for the decorrelating detector (2 users)%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
N=31;
K=2;%%%%no of users
No=0.03;
Nodb=10*log10(1/No)
no_of_bits=100000;
%%%%get signature sequences
load
'/home/users/arun/sscdma/project/mseq31primpoly45.mat'
a1=charphase;
load
'/home/users/arun/sscdma/project/mseq31primpoly75.mat'
a2=charphase;
%%%%%%%generate the anitpodal sequence
a1=2*a1-1;
a2=2*a2-1;
%%%%%%%normalize energy of signature waveforms
a1=a1/sqrt(sum(a1.*a1));
a2=a2/sqrt(sum(a2.*a2));
A=[1 0;0 1];
rho=sum(a1.*a2);
R=[1 rho;rho,1];
bits=round(rand(K,no_of_bits));
b=2*bits-1;
n=sqrt(No)*randn(K,no_of_bits);
y=sign(inv(R)*(R*A*b + n));
%convert to ones and -1s
b_hat=(y+ones(K,no_of_bits))/2;
%get the b.e.r as the average of the 2 users
ber1=sum(xor(bits(1,:),b_hat(1,:)))/no_of_bits;
ber2=sum(xor(bits(2,:),b_hat(2,:)))/no_of_bits;
avg_ber=0.5*(ber1+ber2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%code for the decorrelating detector (10 users)%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
N=31;
```

```
K=10;%%%%no of users
No=0.02;
Nodb=10*log10(1/No)
%No=N*No;
no_of_bits=10000;
A=eye(K);
%get codes and R
load '/home/users/arun/sscdma/project/10goldcodes.mat'

bits=round(rand(K,no_of_bits));
b=2*bits-1;
n=sqrt(No)*randn(K,no_of_bits);


y=sign(inv(R)*(R*A*b + n));
%convert to ones and
b_hat=(y+ones(K,no_of_bits))/2;

ber1=sum(xor(bits(1,:),b_hat(1,:)))/no_of_bits;
ber2=sum(xor(bits(2,:),b_hat(2,:)))/no_of_bits;
avg_ber=0.5*(ber1+ber2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%code for the MMSE linear detector (2 users)%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
N=31;
K=2;%%%%no of users
No=0.1;
Nodb=10*log10(1/No)
%No=N*No;
no_of_bits=1000;
%%%%get signature sequences
load
'/home/users/arun/sscdma/project/mseq31primpoly45.mat'
a1=charphase;
load
'/home/users/arun/sscdma/project/mseq31primpoly75.mat'
a2=charphase;
%%%%%%%generate the anitpodal sequence
a1=2*a1-1;
a2=2*a2-1;

%%%%%%%normalize energy of signature waveforms
a1=a1/sqrt(sum(a1.*a1));
a2=a2/sqrt(sum(a2.*a2));

A=[1 0;0 1];
rho=sum(a1.*a2);
R=[1 rho;rho,1];
novector(1:K)=No;
sigma2Aminus2=diag(novector);

bits=round(rand(K,no_of_bits));
b=2*bits-1;
n=sqrt(No)*randn(K,no_of_bits);

y=sign(inv(R+sigma2Aminus2)*(R*A*b + n));
%convert to ones and minu ones
b_hat=(y+ones(K,no_of_bits))/2;

ber1=sum(xor(bits(1,:),b_hat(1,:)))/no_of_bits;
ber2=sum(xor(bits(2,:),b_hat(2,:)))/no_of_bits;
avg_ber=0.5*(ber1+ber2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%code for the MMSE linear detector (10 users)%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
N=31;
K=10;%%%%no of users
No=0.02;
Nodb=10*log10(1/No)
%No=N*No;
no_of_bits=10000;
%%%%get signature sequences
A=eye(K);
load '/home/users/arun/sscdma/project/10goldcodes.mat'

bits=round(rand(K,no_of_bits));
b=2*bits-1;
n=sqrt(No)*randn(K,no_of_bits);
novector(1:K)=No;
sigma2Aminus2=diag(novector);
```

```
y=sign(inv(R+sigma2Aminus2)*(R*A*b + n));
%convert to ones and
b_hat=(y+ones(K,no_of_bits))/2;

ber1=sum(xor(bits(1,:),b_hat(1,:)))/no_of_bits;
ber2=sum(xor(bits(2,:),b_hat(2,:)))/no_of_bits;
avg_ber=0.5*(ber1+ber2)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%code for the LMS algorithm (2 users)%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
mu=0.01;
N=31;
K=2;%%%%no of users
No=0.08;
Nodb=10*log10(1/No)
%No=N*No;
no_of_bits=100000;
%%%%get signature sequences
load
'/home/users/arun/sscdma/project/mseq31primpoly45.mat'
a1=charphase;
load
'/home/users/arun/sscdma/project/mseq31primpoly75.mat'
a2=charphase;
%%%%%%%generate the anitpodal sequence
a1=2*a1-1;
a2=2*a2-1;

%%%%%%%normalize energy of signature waveforms
a1=a1/sqrt(sum(a1.*a1));
a2=a2/sqrt(sum(a2.*a2));

A=[1 0;0 1];
rho=sum(a1.*a2);
R=[1 rho;rho,1];

bits=round(rand(K,no_of_bits));
b=2*bits-1;
n=sqrt(No)*randn(K,no_of_bits);

y=(R*A*b + n);
%%%%%%%%%%%theoretical optimum weights
Wopt=inv(R*A*A*R+No*R)*[1;rho];

c1=[1,1]';
mean_squared_error(1)=sum((Wopt-c1).^2)/K;
c1=c1-mu*(c1'*y(:,1)-b(1,1))*y(:,1);
w1(1)=c1(1,1);
w2(1)=c1(2,1);

for i=2:no_of_bits
   %c1=c1-mu*(0.995)^(i-1)*(c1'*y(:,i)-b(1,i))*y(:,i);
        %%%variable step size
   c1=c1-mu*(c1'*y(:,i)-b(1,i))*y(:,i);%%%%fixed step
size
   w1(i)=c1(1,1);
   w2(i)=c1(2,1);
   mean_squared_error(i)=sum((Wopt-c1).^2)/K;
end
Wopt
bits_hat=sign(c1(1)*y(1,:)+c1(2)*y(2,:));
b_hat=(bits_hat+ones(1,no_of_bits))/2;
ber1=sum(xor(bits(1,:),b_hat(1,:)))/no_of_bits
figure(3);
subplot(2,1,1);
plot(w1);
subplot(2,1,2);
plot(w2)
figure(4)
plot(mean_squared_error);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%code for the blind MUD algo. (2 users)%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
mu=0.0005;
N=31;
K=2;%%%%no of users
No=0.1;
Nodb=10*log10(1/No)
no_of_bits=50000;
%%%%get signature sequences
```

```
load
'/home/users/arun/sscdma/project/mseq31primpoly45.mat'
s1=charphase;
load
'/home/users/arun/sscdma/project/mseq31primpoly75.mat'
s2=charphase;
%%%%%%%generate the anitpodal sequence
s1=2*s1-1;
s2=2*s2-1;
%%%%%%%normalize energy of signature waveforms
s1=s1/sqrt(sum(s1.*s1));
s2=s2/sqrt(sum(s2.*s2));

bits=round(rand(K,no_of_bits));
b=2*bits-1;

n=sqrt(No)*randn(K,no_of_bits);
x1(1,1:N)=0;
for i=1:no_of_bits
   transmitted_bits=b(1,i)*s1+b(2,i)*s2;
   y=transmitted_bits+sqrt(No)*randn(1,N);%%received
                                          %%statistic
   zmf=sum(y.*s1);
   z=sum(y.*(s1+x1(i,:)));
   x1(i+1,:)=x1(i,:)-mu*z*(y-zmf*s1);
   x1(i+1,:)=x1(i+1,:)-sum(s1.*x1(i+1,:))*s1;%replace by
                           %the orthogonal projection
   %orth(i)=sum(s1.*x1(i+1,:));
   b_hat(i)=sign(z);
sir(i)=1/(No*(1+sum(x1(i+1,:).*x1(i+1,:)))+(sum((s1+x1(i
                                  +1,:)).*s2))^2);
end
b_hat=(b_hat+ones(1,no_of_bits))/2;
ber1=sum(xor(bits(1,:),b_hat(1,:)))/no_of_bits

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%code for the blind MUD algo. (10 users)%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
mu=0.005;
N=31;
K=10;%%%%no of users
No=0.1;
Nodb=10*log10(1/No)
no_of_bits=1000;
load 10goldcodes.mat
bits=round(rand(K,no_of_bits));
b=2*bits-1;

n=sqrt(No)*randn(K,no_of_bits);
x1(1,1:N)=0;
for i=1:no_of_bits
   transmitted_bits=zeros(1,N);
   for k=1:10
      transmitted_bits=b(k,i)*u(k,:)+transmitted_bits;
   end
   y=transmitted_bits+sqrt(No)*randn(1,N);%%received
                                          statistic
   s1=u(1,:);
   zmf=sum(y.*s1);
   z=sum(y.*(s1+x1(i,:)));
   x1(i+1,:)=x1(i,:)-mu*z*(y-zmf*s1);
   x1(i+1,:)=x1(i+1,:)-sum(s1.*x1(i+1,:))*s1;%replace by
the orthogonal projection
   orth(i)=sum(s1.*x1(i+1,:));
   b_hat(i)=sign(z);
   mai=0;
   for k=2:10
      mai=mai+sum((s1+x1(i+1,:)).*u(k,:));
   end
   sir(i)=1/(No*(1+sum(x1(i+1,:)))+mai)^2;
end
b_hat=(b_hat+ones(1,no_of_bits))/2;
ber1=sum(xor(bits(1,:),b_hat(1,:)))/no_of_bits
```

REFERENCES:

[1] R.L.Peterson, R.E.Ziemer and D.E.Borth, *Introduction to Spread Spectrum Communications*, Prentice Hall, Inc., 1995.

[2] Tan F. Wong, *Spread Spectrum and CDMA* :Class notes, Fall 2001.

[3] S. Verdu, *Multiuser Detection*, Cambridge University Press, 1998.

[4] J.G. Proakis, *Digital Communications*, 3rd Ed., McGraw-Hill, Inc., 1995.

[5] M. Honig, U. Madhow, and S. Verdu, ``Blind Adaptive Multiuser Detection," *IEEE Trans. Inform. Theory*, vol. 41, pp. 944-960, Jul. 1995.

ACKNOWLEDGEMENTS

I would like to thank Dr.Tan F. Wong for useful discussions that directed my simulations and helped me understand the material presented in this report. I would also like to thank my friend Yadu Rao for helping with the matrix calculus involved in section III.