# INVESTIGATION OF AN ITERATIVE GROUPWISE SOFT INPUT/SOFT OUTPUT MULTIUSER DETECTION ALGORITHM

by

Bradley Zarikoff
Bachelor of Engineering, University of Victoria, 2002

THESIS
SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

In the School
of
Engineering Science



SIMON FRASER UNIVERSITY

June 2004

# APPROVAL

**Name:** **Bradley Zarikoff**

**Degree:** **Master of Applied Science**

**Title of Thesis:** Investigation of an Iterative Groupwise Soft Input/Soft Output Multiuser Detection Algorithm

**Examining Committee:**

**Chair:** **Dr. John D. Jones**
Associate Professor of the School of Engineering Science

---

**Dr. James K. Cavers**
Senior Supervisor
Professor of the School of Engineering Science

---

**Dr. Paul K. M. Ho**
Supervisor
Professor of the School of Engineering Science

---

**Dr. Dong In Kim**
Internal Examiner
Associate Professor of the School of Engineering

**Date Defended/Approved:** June 18, 2004

# SIMON FRASER UNIVERSITY

## Partial Copyright Licence

# ABSTRACT

This thesis investigates the performance of a novel iterative soft input/soft output groupwise multiuser detector (MUD). In general, multiuser detectors improve performance by explicitly accounting for the interference when detecting all users together. In this thesis, the interference is assumed to be a product of users whose channel gains are known. The novel technique extends the group detector MUD with an iterative process, and is aptly named iterative MUD (IMUD). The success of IMUD is measured by how close it can get to the performance of the computationally prohibitive optimum technique (joint maximum likelihood MUD) with as little complexity as possible. To do this, IMUD was compared to existing MUDs in terms of complexity and performance. Since MMSE V-BLAST was seen as the best low complexity competitor, it is used as a performance reference.

The results from IMUD with equal numbers of users and receivers were excellent. In a 12 user/12 receiver system at an error rate of $10^{-3}$, IMUD had a 2dB gain on MMSE V-BLAST and roughly equal complexity. In the case when there are more users than receivers, IMUD's gains are even more pronounced. In a 12 user/8 receiver flat fading system at an error rate of $10^{-3}$, IMUD had a 7dB gain on MMSE V-BLAST with roughly equal complexity. IMUD was also tested in a serially concatenated system in which all users employed convolutional codes. The results are promising. In a 6 user/4 receiver system at an error rate of $10^{-3}$, the IMUD/SISO detector performs within 2dB of a single user coded system with the same number of antennas. With a rate 2/3 code, IMUD/SISO performs within less than 2dB of a single user coded system.

# ACKNOWLEDGEMENTS

This work has been carried out in the Mobile Communications Laboratory at Simon Fraser University, Burnaby, BC and with Group Research A at Tait Electronics Ltd., Christchurch, NZ.

My supervisor was Dr. James K. Cavers, whose advice and guidance were invaluable throughout the research and thesis writing process. He showed me that iterative techniques really do improve performance. Thanks to the entire MTF lab for being a great place to sound ideas.

Kudos to Tait Electronics for the incredible open and friendly research environment they have. Cheers to the entire Group Research crew. Many thanks as well to Des Taylor and Philippa Martin for the discussions on iterative techniques.

Finally, thanks to my family; my parents Betty and Bill, my brother Peter, and my sisters Dina and Michele, and my partner Emillie, for supporting me throughout.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1 INTRODUCTION

Communications Engineering is focused on the transmission of information from point A to point B. For the past century, this transfer of information has become increasingly important in society. Communications research has always been driven by current applications: initially to communicate between cities and colonies separated by vast oceans and now for mobile voice, data, and video on demand. These new applications depend on high bandwidth systems. "High-bandwidth systems" usually denotes systems with significantly increased capacity compared with existing systems, allowing new applications to flourish. The digital revolution sparked this trend of catch-up, where engineers are constantly evolving current systems to meet tomorrow's demands. Integrated circuits and micro-processors have been a key catalyst in allowing designers to use complex algorithms in communications systems. These tools continue to allow awkward and complex theoretical communications systems that are born in universities and research labs to become reality.

A solution to the high bandwidth problem is under investigation by the engineering community, but there are many complications. One of these complications is defined by society, not by engineering principles. The popularity of wireless systems has made the electromagnetic spectrum a hugely popular commodity. Hence, the engineer is constantly looking for ways to compress more information into smaller and smaller allocations of spectrum while keeping the integrity of the communications system intact. The term 'capacity' is used by engineers to refer to the amount of information transferred in a given bandwidth. It is measured in bits/second/Hertz, and can be thought of as a normalized data rate, or a throughput for a certain bandwidth. To put it

concisely, the study of how to maximize the capacity of a communications system while keeping within a finite bandwidth is the result of our need for high bandwidth systems. This is one of the primary goals of communications research today.

Multiple antenna systems (commonly referred to as multiple-input multiple-output, or MIMO) are one of the techniques devised to address this problem of capacity. By using statistical properties of the wireless channel, multiple antennas can multiply the capacity of the original single antenna system. The trade-off comes in the form of physical space and the price of adding additional antennas versus that of bandwidth.

Multiuser detection (MUD) is a subset of MIMO. In MUD systems, the users accessing the channel are allowed to interfere with each other. In a way, this interference can be used to increase the capacity of the system by placing more information in the same mobile channel. The detection of such a signal seems to be quite a challenge, given that the symbols become apparently irreversibly combined. However, due to the nature of the mobile channel, such a system has been proven to be practical. The variety in the channel that each user experiences actually provides the information necessary to distinguish each user from each other. Without the diverse and variable nature of the mobile channel, MUD would not be possible. For example, in a line-of-sight satellite communications link, MUD with identical pulse shapes is not an option because of the lack of multipath.

The technique documented in this thesis is a novel integration of various existing multiuser techniques. It improves on a groupwise technique that bridges the optimal and suboptimal techniques, allowing design engineers to trade off performance for computational complexity. The technique, called iterative MUD (IMUD), is novel in the way that it allows iterations between groups, and not in time as with most iterative processes. This iterative process proves to be of great value in increasing the

performance with a linear increase in computations as opposed to the exponential increase apparent in the optimal technique. In fact, in an overloaded system that contains more users than receivers, the performance gains are even greater. Most other suboptimal techniques perform very poorly in this case because of their heavy reliance on linear estimation.

Since the new MUD technique is effectively a soft-input/soft-output (SISO) block, it is also examined as part of a serially concatenated iterative system. Convolutional codes are used to provide channel memory and allow iteration of the calculated probabilities.

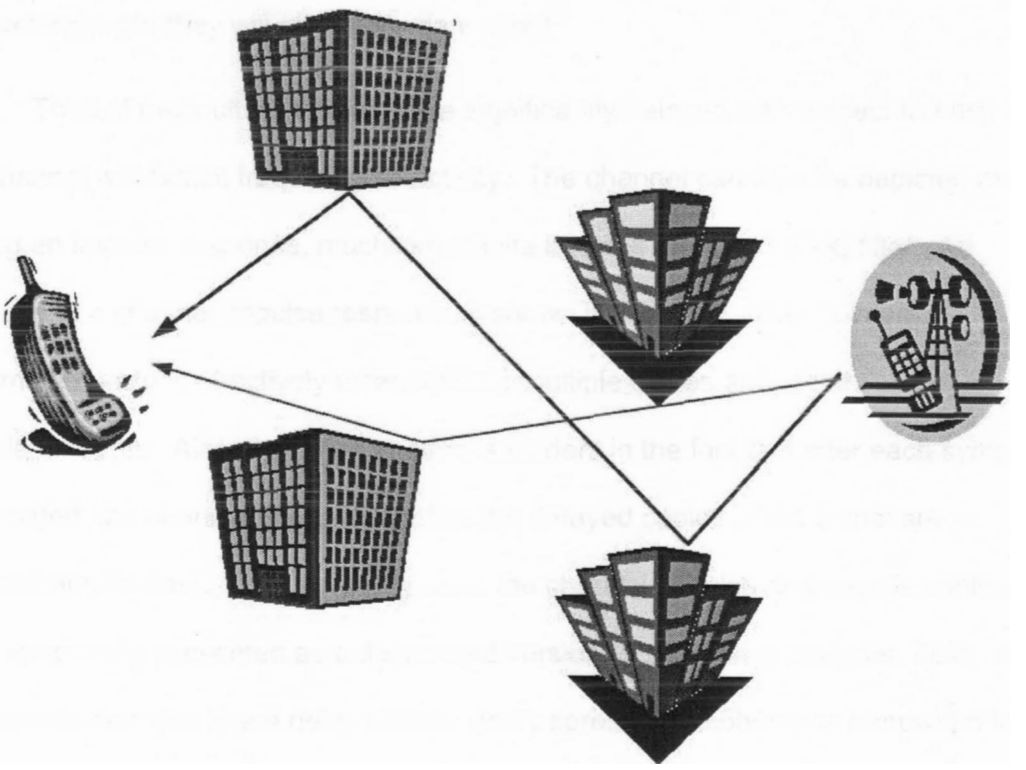
# 2 BACKGROUND

## 2.1 The Mobile Channel

In this thesis, the term 'user' refers to a mobile transceiver and the term 'receiver' refers to a stationary transceiver. This study focuses on data transmission from the user to the receiver, also called the uplink, although in general the techniques described in the following chapters are viable for both the user and the receiver. A likely scenario could include a cellular phone customer, where the user is the handset and the receiver is the base station.

A mobile channel links the user and receiver. The term 'mobile channel' is used to refer to a radio frequency channel that changes over time and distance. The mechanism behind the variation in the mobile channel is illustrated in Figure 1. The channel includes multiple reflected radio wave paths between the user and the receiver and either a weak or non-existent direct path. As the user moves relative to the scatterers (the reflective objects), all channel paths have the potential to change. Hence, a mobile communications system must be able to operate using only the reflections of the signal, collectively called multipath. The first step in dealing with multipath is to identify the state of the channel at each sampling time. The state of a channel is a mathematical value which represents the amplitude change and phase shift that a signal would experience. This is also commonly known as the channel state information (CSI). The CSI is used along with a specific detection scheme to reverse the effects of the multipath, allowing for effective reception of the transmitted signal.

**Figure 1: Multiple paths in a mobile communications system**



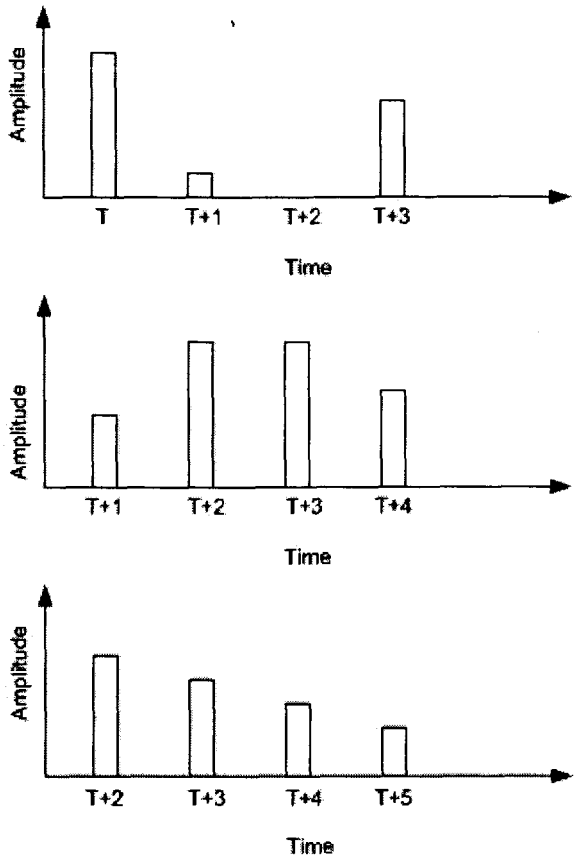
A communications system operating in a mobile channel has to deal with a number of detrimental effects. First, since the multipath channel is constantly changing due to movement of the user or the scatterers, the channel is time-variant. The variability depends on the nature of the channel; in a busy downtown core with the mobile located in a moving vehicle, the channel changes frequently, but with the mobile located with a pedestrian on foot in the countryside, the channel may be static. In the case of the frequently changing channel, the communications system must track the CSI as the channel varies.

Second, there is the possibility of a fade due to the reflections. A fade occurs when the signal strength or amplitude is reduced to below a useable threshold, where the usability is defined by an error rate. Fades are due to multipath with relatively equal

amplitudes arriving at the receiver out of phase with each other. If they are close to half a wavelength off, they will effectively cancel out.

Third, if the multipath signals are significantly delayed with respect to each other, the channel will exhibit frequency selectivity. The channel can then be depicted as having an impulse response, much like a finite impulse response (FIR) filter. An example of a channel impulse response is shown in Figure 2. Note how each transmitted pulse is effectively extended into multiple copies arriving at subsequent sample intervals. Also, the time variation is evident in the fact that after each symbol is transmitted, the channel is altered (notice the delayed copies of the signal are at different amplitudes). In an actual system, the channel impulse response is continuous, but it is normally presented as a discretized version to allow easy analysis. Many terms are used to describe these delay effects; delay spread and coherence bandwidth for example. Delay spread defines the average length of time it takes for all multipath signals to arrive at the receiver. In Figure 2, the average delay spread is 3 ½ symbols. Coherence bandwidth defines the bandwidth in which the channel response is highly correlated. The bandwidth can be used as a tool to reduce the delay spread in the system. If the signal is bandlimited to within the coherence bandwidth, the problem introduced by this signal spread may be averted, but at the cost of capacity. Making use of the Fourier transform, the delay spread is the inverse of the coherence bandwidth.

**Figure 2: Example of delay spread and channel variance over time**




Symbols transmitted at times T, T+1 and T+2

So far the mobile channel can vary over time and exhibit frequency selectivity. Another aspect of time variation that is of importance is Doppler spread. Doppler refers to signal spread over the frequency band due to the relative velocity between the transmitter, the scatterers and the receiver. If the mobile channel is non-varying, there is no Doppler spread. If the mobile channel varies, the Doppler spread effectively spreads the transmitted signal by an amount determined by the Doppler frequency. The inverse of the Doppler frequency is called the coherence time, and is the best way to view the effects of the Doppler shift. The coherence time is the amount of time that the channel stays relatively correlated. If the coherence time is less than a symbol time, than there is a possibility that a fade may occur within the transmission of a symbol. The traditional

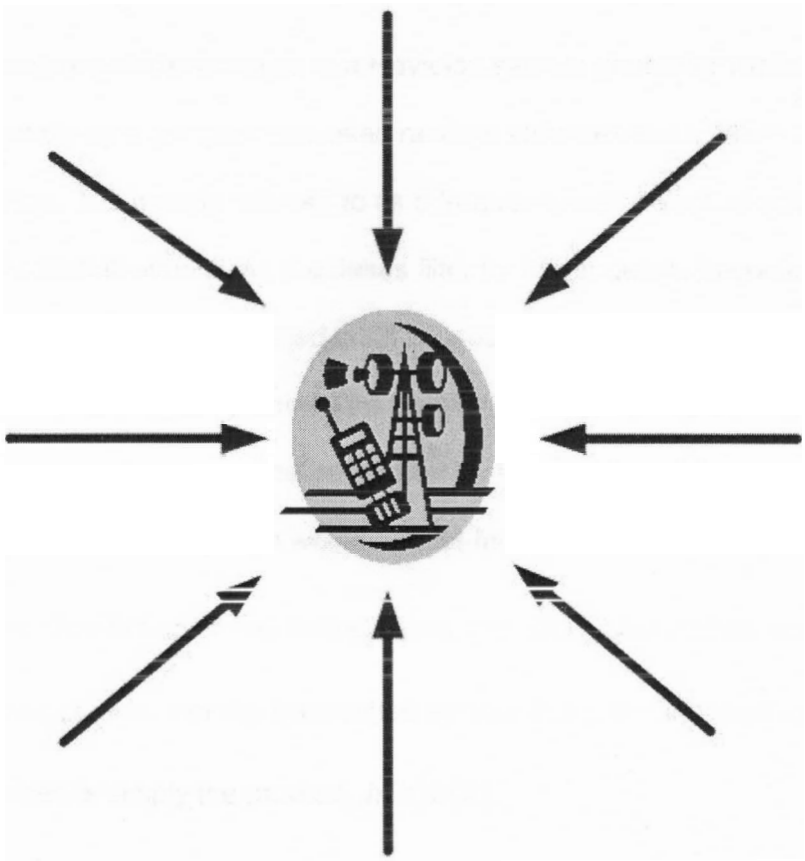
expression for Doppler frequency is $f_d = \frac{vf_0}{c}$, where $v$ is the velocity of the user, $f_0$ is the carrier frequency, and $c$ is the speed of light. For the mobile channel, it is not as simple since the spread is due to a number of contributors, including the movement of other users and scatterers. In any case, this is only a brief description of the characteristics of the mobile channel. Our simulations will be making significant simplifications on the system. For a thorough discussion of multipath and all of its components, see [1].

For the study of the iterative multiuser detector, the simulated channel is relatively simple. The simulations take place in a channel that has no delay spread. Since the point of the simulations of the new multiuser detector is to determine how it performs relative to other detectors, the more complex channels are avoided for now. As justification, the coherence bandwidth of the system is assumed to be much greater than the signal bandwidth. The response of the system then becomes a single impulse, eliminating the delay spread. A system with no delay spread is termed flat-fading since there is no frequency selectivity.

Various ways to model the flat-fading multipath channel have been studied and utilized in real systems. They all consist of expressing the channel gain experienced by the transmitted pulses as a single random variable. The most common for a non-line-of-sight system is the Rayleigh model. The Rayleigh model assumes that the incoming signals are numerous (Figure 3). All signals are assumed to have random phase. The Rayleigh model can be derived from a complex Gaussian model. In the mobile channel, the numerous received signals allow a complex Gaussian approximation to be made [1]. The probability distribution function (pdf) of the complex Gaussian amplitude turns out to be Rayleigh. A key feature is that the Gaussian assumption holds up even for less than 10 incoming signals, allowing even simple mobile channels to match this model. Of

course, this model is a major approximation. However, field tests have shown that these approximations generate channels with statistics that come very close to those of a measured mobile channel. Thus, we use the Rayleigh fading channel in our simulations.

**Figure 3: Equispaced incoming signals for the Gaussian assumption**



Our model now consists of two important points; our channel is Rayleigh fading with no delay spread. However, this information tells us nothing about the time variation of the channel. For that, we use the Jakes model [1,2] to provide the time correlation introduced by movement of the user and scatterers. The Jakes model forces the correlation in time of the channel gains to match that of a channel with a specific Doppler frequency. This Doppler frequency is a product of the movement of the user and all the scatterers in the channel. For example, if the Doppler frequency were set to zero, the

channel would be static, and if the Doppler were set to $\infty$, the channel would be uncorrelated from one instant to the next. The model makes the assumption that the autocorrelation of the channel impulse response is a zeroth order Bessel function, $J_0(2\pi f_d T)$. The product of Doppler frequency and symbol duration, $f_d T$, is used to set the rate of channel change. A common value for a fast-fading channel is $f_d T = 0.01$.

Our complete model consists of a Rayleigh sample generator (which can be represented simply by a complex Gaussian random variable) and a Jakes filter to model the time variation. It is usually referred to as a frequency-non-selective slow-fading channel. In the Matlab simulation, the Jakes filter for larger delays becomes excessively lengthy. So, the technique as detailed in [2] is used, which allows for speedy computation. This technique combines the Jakes filter with a complex Gaussian sample generator. If we were to remove the Jakes filter so that the samples are uncorrelated from symbol to symbol, our system would be fast-fading.

In either slow-fading or fast-fading cases, the gain is denoted as $h(k)$, where $k$ denotes the symbol time. For the transmitted symbol $b(k)$, the received symbol prior to noise being added is simply the product, $h(k)b(k)$.

In a multiple-input, multiple-output (MIMO) antenna system, we have a number of transmitted signals and we receive them with a number of antennas. If the antennas are sufficiently separated and if the users are sufficiently separated, the various Rayleigh channels are uncorrelated. Usually a half wavelength separation is enough to provide decorrelation at the mobile. At the base station, a significantly larger separation is normally required. However, except for some small oscillations in the autocorrelation function, the further apart the antennas, the more decorrelated the paths. The independence of the channels is needed for channel diversity. Diversity is a term used

to convey the fact that the signals arriving at different antennas experience difference depths of fade; the higher the diversity of a system, the higher the amount of information available at the receiver about the transmitted symbols

Diversity and antenna separation can be intuitively seen in the following example. If the scattered signals are reflected close to the receiver, thus having a large angle of separation between the signals, then a small change in the location of the receiver will result in a large change to all the channels. Conversely, if the reflections occur from a far distance, creating a small angle of separation between the signals, the same change in location may result in virtually no change to the channel gains. The angle of arrival is the important factor.

To reiterate, the effect of delay spread and the resulting interference from neighbouring symbols will not be included in the investigation of IMUD. We have decided to leave this for further study, and focus on the effects of our novel detector in a relatively uncomplicated wireless channel. However, it is worth noting that delay spread can be seen as a bonus in a mobile system. If the delayed symbols are detected in the optimal sense, they can be viewed as extra information for the past symbols arriving with the current symbols. One transmitted symbol is therefore spread between multiple channels, where these channels are separated in time instead of distance as is the case with MIMO. Thus, it would increase the dimensionality of our system in a beneficial way at the cost of the optimum detector complexity. The same argument could be made for systems that use spreading codes or asynchronous transmission.

The simulations for IMUD are done with a frequency non-selective, fast-fading channel model. Since the multiuser detector operates on a symbol-by-symbol basis, any change to the correlation in channel samples will leave the simulation results unchanged. This fact allows the simulations to be run with totally uncorrelated channels

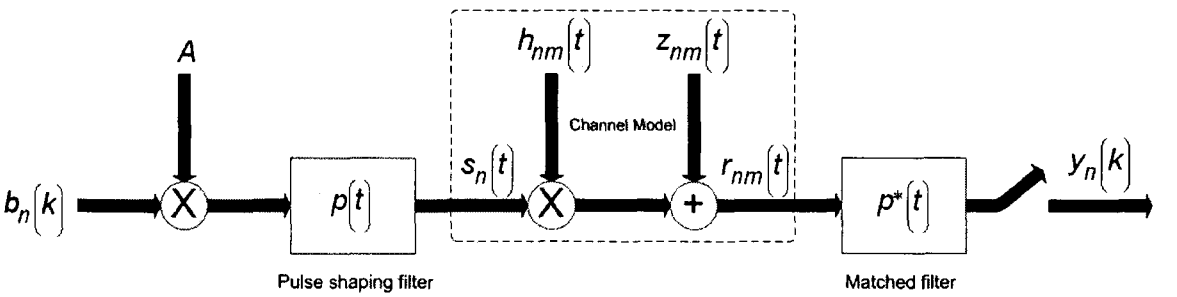
(each symbol experiences a channel gain that is uncorrelated to the last). This also allows the results for IMUD to be generalized to any type of correlated channel, fast or slow. However, whether the channel is fast-fading or slow-fading does affect performance of a system that combines IMUD with memory dependent schemes, such as channel coding.

## 2.2 Communications System

For our investigation, we assume that we have $N$ users (or $N$ antennas at a transmitter array, see Figures 5 and 6) that are synchronous and transmitting independent data symbols. Each transmitter sends a binary phase shift keyed (BPSK) signal. The signals are received with equal power at an $M$ element receiver array.

The notations used in this thesis are as follows: an italicized lower case character denotes a scalar and may be indexed by time (ie. $b(k)$), a bold and lower-case character denotes a vector (ie. $\mathbf{x}$, $\mathbf{x}(k)$ if time indexed), a bold and upper-case character denotes a matrix (ie. $\mathbf{X}$), $\mathbf{X}^{\mathrm{T}}$ denotes transpose, $\mathbf{X}^{\dagger}$ denotes a conjugate transpose, and $\mathbf{X}^{\mathrm{P}}$ denotes the pseudo-inverse. $\mathbf{I}_M$ is an $M \times M$ identity matrix

**Figure 4: SISO channel model**



From now on, the independent complex Gaussian random variable $h(t)$ will be denoted for all $M \times N$ MIMO channels by the $M \times N$ time varying channel matrix $\mathbf{H}(t)$. $\mathbf{H}(t)$ is also represented as $N$ column vectors, $\mathbf{H}(t) = \left[\mathbf{h}_1(t)\ \mathbf{h}_2(t)\ \cdots\ \mathbf{h}_N(t)\right]$, where $\mathbf{h}_i(t)$ is an $M \times 1$ column vector that represents the channel gains from user $i$ to all $M$ receiver antennas. The receiver noise is shown as the length $M$ vector $\mathbf{n}(t)$. Finally, the transmitted symbols from all $N$ users will be shown as $\mathbf{b}(k) = \left[b_1(k) \cdots b_N(k)\right]^T$, which is a length $N$ vector consisting of BPSK symbols belonging to the set $\{-1, +1\}$. The received signals are matched filtered with $p(t)$, which is the transmit pulse shape. After sampling at the receiver, the channel gains and noise are denoted as $\mathbf{H}(k)$ and $\mathbf{n}(k)$, respectively. They are stored in the length $M$ vector $\mathbf{y}(k)$. $\mathbf{y}(k)$ is expressed as

$$\mathbf{y}(k) = \mathbf{H}(k)\mathbf{b}(k) + \mathbf{n}(k) \tag{1}$$

We have assumed symbol synchronous reception and no inter-symbol interference. For notational simplicity, we will neglect the discrete time reference on the vectors; $\mathbf{H}(k)$, $\mathbf{b}(k)$, and $\mathbf{n}(k)$ become $\mathbf{H}$, $\mathbf{b}$, and $\mathbf{n}$, respectively. For the simulations involving convolutional codes and interleaving, the time dependency is assumed in the calculations. However, for the investigation of IMUD alone, the fact that IMUD is strictly symbol-by-symbol allows the channel gain to be modelled as independent from one symbol to the next. This helps speed up simulation time. Equation (1) then becomes

$$\mathbf{y} = \mathbf{H}\mathbf{b} + \mathbf{n} \tag{2}$$

The signal-to-noise ratio (SNR) of the single link is defined as follows. First, we use the knowledge that the system is symbol synchronous and flat-fading to allow each pulse to be considered separately. Each transmitted symbol is scaled by a value $A$. $b_n$

is then pulse shaped with $p(t)$ before being transmitted. $p(t)$ is set equal to a Nyquist pulse with unit energy for simplicity ( $\int |p(t)|^2 dt = 1$). Each user transmits from a single antenna. For the channel from the $n^{th}$ user to the $m^{th}$ receive antenna, the transmitted signal power can be found by taking the expected value of the square of the transmitted signal. The symbols are transmitted at rate of $1/T$ through the channel shown in Figure 4. $h_{nm}$ and $z_m$ are complex Gaussian random variables with variance equal to ½ and $N_0$, respectively. $b_n$ is then pulse shaped with $p$ before being transmitted. For the channel, we set $\mathrm{E}\left[|h_{nm}|^2\right] = 1$ so that the variance of $h_{nm}$ is equal to ½ as noted above. Then the received signal energy is

$$\begin{aligned} \mathrm{E}_{s_n} &= \frac{1}{2} \cdot \mathrm{E}\left[|s_n h_{nm}|^2\right] \\ &= \frac{1}{2} \cdot \mathrm{E}\left[|s_n|^2\right] \cdot \mathrm{E}\left[|h_{nm}|^2\right] \\ &= \frac{1}{2} \cdot A^2 \cdot 1 = \frac{A^2}{2} \end{aligned} \tag{3}$$

The received signal is $y_m = A \cdot h_{nm} b_n + n_m$, where $n_m$ is the filtered noise term. Since we sample at a rate of $1/T$, the variance of $n_m$ is identical to the variance of $z_m$. The received SNR from the $n^{th}$ user to the $m^{th}$ antenna is

$$\mathrm{SNR} = \frac{\mathrm{E}_{s_n}}{\frac{1}{2}\mathrm{E}\left[|n_m|^2\right]} = \frac{\frac{1}{2}A^2}{N_0} \tag{4}$$

For the rest of the document, $A$ is set to unity and the SNR is based solely on the noise power $N_0$.

Our multiuser MIMO system is depicted in Figure 5. An alternative system which has identical capacity is shown in Figure 6. It makes use of uncorrelated antennas at the same transmitter, and achieves capacity gains by transmitting multiple statistically independent streams.

**Figure 5: Multiuser system (MIMO)**

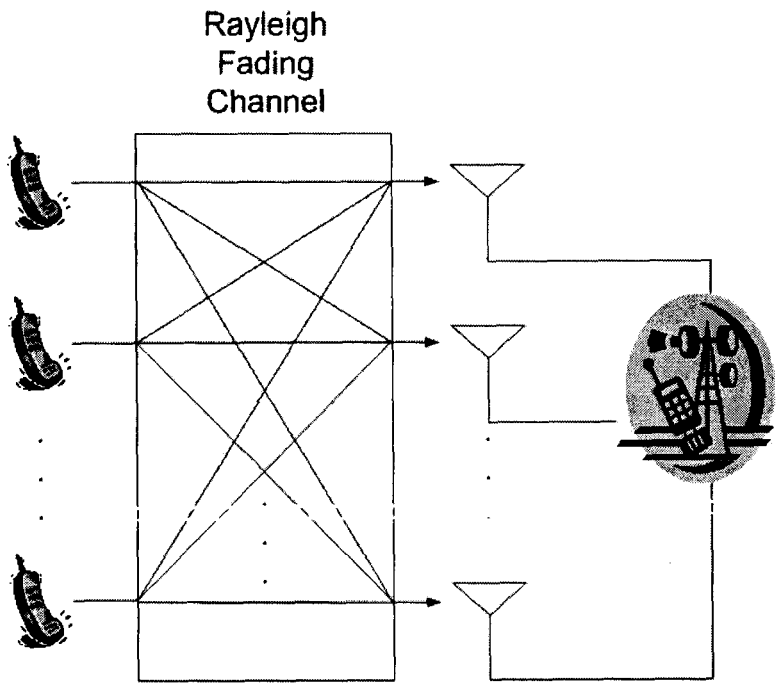
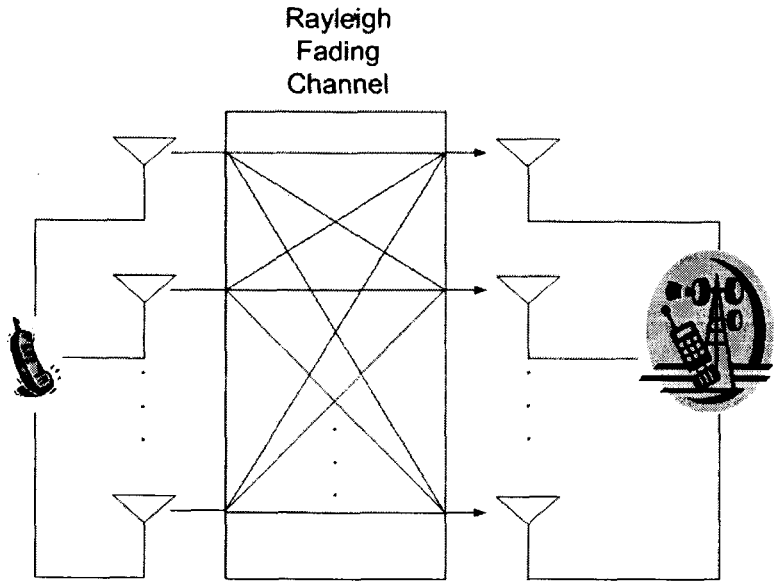


**Figure 6: Multistream system (MIMO)**




## 2.3 Single-User Detection Techniques

Traditional mobile systems with a single transmitter and receiver use a single-user detector with optimum results. In some multiple access techniques, such as time division or frequency division, the pulse shapes of each user are made orthogonal. This also allows for the use of a single-user detector. Without multiple interfering users, there is no need for multiple-user detection. However, there is still a possibility of past symbols interfering with current symbols; channels with delay spread are prime examples. But this is traditionally dealt by with equalizers, which use the knowledge of the channel's impulse response to remove the past symbols' effects.

In a system with no delay spread, the single-user detector is greatly simplified. The optimum detector, given that all symbols are equi-probable, can be realized as the correlator receiver. This detector uses a method of ranking all possible transmitted symbols by the correlation between each possible symbol and the received signal. It is shown [1] that the correlation metric can be found as

$$D(y,b_k)=|y-hb_k|^2 \tag{5}$$

where $y$ is the received sample, $h$ is the channel gain. $b$ is the hypothesized symbol sent and is one of $m$ values for an $m$-ary constellation. This is derived for a known channel with Gaussian noise. A convenient way of viewing this technique is that of a 'closest match'. $D$ is essentially a distance metric between the received sample and the current hypothesis. Whichever hypothesis gives the smallest distance is the decision. In other words, $\hat{b}$, the estimated symbol, is $\arg\min_b D(y,b)$. With that in mind, we can view the detection via a plot of the constellation points and the received sample. The distance between each constellation point is simply bisected by a line, denoting the decision line. For BPSK, with $\{+1,-1\}$ as possible symbols, this decision line is simply the y-axis. If our sample lies on the negative side of the x-axis, we decide the transmitted symbol was –1, and vice versa.

The rationale for the correlator receiver comes from the fact that the noise that corrupts our transmitted signal is Gaussian. The metric falls from the likelihood function of $y$ given a hypothesis $b$. Putting this in context of our communications system shown in the previous section, (2) is first reduced for a single user as $y=hb+n$. Since $n$ is assumed to be Gaussian, and we have a hypothesis $b_k$ and channel measurement $h$, we can easily find $\mathrm{p}(y\mid h,b)$ as

$$\mathrm{p}(y\mid h,b_k)=\frac{1}{2\pi\sigma^2}e^{-\frac{|y-hb_k|^2}{2\sigma^2}} \tag{6}$$

since the mean of $y$ conditioned on $b$ is $hb$ and the variance of $n$ is $\sigma^2$. Since the symbols are equi-probable (maximum likelihood decisions), the pdf scaled with respect to $b$ is equivalent to $\Pr(b)$, which is the probability of a given symbol. Taking the natural

log of this pdf results in a constant plus the distance metric in (5). Since the logarithm function is an invertible operation, the symbol that has the lowest value for (5) can be said to have the probability $\Pr(b)$.

This optimum technique changes if any *a priori* information is available about the symbols. The above formulation assumes that all possibly transmitted symbols are equi-probable. If they aren't, Bayes rule must be used to transform the above maximum likelihood (ML) technique into a maximum *a posteriori* (MAP) technique. The probability of the current hypothesis becomes $\Pr(b_k)$

$$\Pr(b_k \mid y) = \frac{\mathrm{p}(y \mid h, b_k)\Pr(b_k)}{\mathrm{p}(y)} \tag{7}$$

where $\Pr(b)$ is the *a priori* information and $\mathrm{p}(y)$ is for normalization.

It's worth noting that for the single-user detector, the complexity is limited to the size of the constellation.

## 2.4 Multiple User Detection Techniques

The optimum multiuser detection (MUD) technique is similar to the optimum single user detection technique, except the size of the constellation for the distance measure is proportionally larger to the number of users. We look at the extension to the single user detection in joint ML detection. A number of sub-optimum techniques are also examined due to their lowered complexity.

### 2.4.1 Joint Maximum Likelihood Detection

Joint maximum likelihood detection (JML) was first detailed by Verdu in the context of different user pulse shapes in [3]. In [4], Grant et al. analyzed JML in the

context of identical pulse shapes and developed a bound for this case. We focus on this paper since in our investigations we use identical pulse shape for all users.

JML makes a closest fit decision given the received samples from all $M$ antennas and channel gains for all $MN$ channels. It can be viewed as a multi-user equalizer, where all users are separated out in a similar fashion in the ML sequence estimator (MLSE). Just as MLSE in the presence of delay spread expands the constellation to deal with the past symbols and their channel gains, JML expands the constellation to encompass all users and their channel gains. An expression for the $M \times 1$ received sample vector $\mathbf{y}$ is given in (2). As with the single user ML, the Gaussian model is assumed for the noise at the $M$ antennas. $\mathbf{y}$ has a mean of $\mathbf{Hb}$ and a variance of $\mathbf{R}_U$. The $U$ stands for 'undesired', and for now it will be taken to be noise. Considering that the variables are complex, the pdf of $\mathbf{y}$ conditioned on the channel gains and the transmitted symbols is

$$p(\mathbf{y} \mid \mathbf{H}, \mathbf{b}) = \frac{1}{2\pi |\mathbf{R}_U|} e^{-\frac{1}{2}(\mathbf{y}-\mathbf{Hb})^{\dagger} \mathbf{R}_U^{-1} (\mathbf{y}-\mathbf{Hb})} \tag{8}$$

If the undesired component of $\mathbf{y}$ is spatially white noise, then $\mathbf{R}_U = N_0 \mathbf{I}_M$. Then $|\mathbf{R}_U|$ becomes $NN_o$ and $\mathbf{R}_U^{-1}$ becomes $N_0^{-1} \mathbf{I}_M$.

The likelihood function (8) is now evaluated for every possible combination of $\mathbf{b}$. Just as in single user, (8) can be represented more conveniently in log form, and is referred to as the JML metric

$$\hat{\mathbf{b}} = \underset{\mathbf{b}}{arg\,min} (\mathbf{y} - \mathbf{Hb})^{\dagger} \mathbf{R}_U^{-1} (\mathbf{y} - \mathbf{Hb}) \tag{9}$$

Since $\log(2\pi |\mathbf{R}_U|)$ is a constant for all $\mathbf{b}$, it is disregarded in the metric.

A bound for the performance of JML from [4] is used in the simulations. It is shown that for the $n^{th}$ user, the union bound of the probability of bit error is

$$P_{s_m} \leq \sum_i \frac{1}{\left(1-\frac{p_{ij2}}{p_{ij1}}\right)^{2M-1}} \sum_{k=0}^{M-1} \binom{2M-1}{k} \left(-\frac{p_{ij2}}{p_{ij1}}\right)^k \tag{10}$$

where $p_{ijk}$ is the $k^{th}$ pole of the characteristic function of a quadratic form, $p_{ij1}$ is in the left-hand plane, $p_{ij2}$ is in the right-hand plane, and $i$ is taken over all possible error vectors ($i$ is incorrect and $j$ is correct **b** vector).

The complexity of implementing this algorithm is reliant on the size of **b**, and therefore the number of users. Using $m$-ary signalling and $N$ users, our complexity grows in proportion to $m^N$. For larger constellations and a few users, this quickly grows out of the computational limits for current systems. For example, with 16QAM and 8-user system, we will have to calculate the metric (9) for greater than four billion combinations of **b**!

### 2.4.2 Joint Maximum A Posteriori Detection

Joint maximum *a posteriori* detection (JMAP) incorporates any previous knowledge with the metric used in JML. This is similar to the MAP detection explained in the single user techniques previously. Again, just as in the single user case, if there is no *a priori* information (the incoming symbols are assumed to be equi-probable) then JMAP becomes JML.

Instead of a searching for a metric to sort the likelihood of each possible vector, JMAP seeks the probability, $\Pr(b_n \mid \mathbf{y})$ for the $n^{th}$ user. The hypothesis with highest probability is then chosen as the decision. JMAP calculates this conditional expression;

the probability of the $n^{th}$ user symbol conditioned on the received samples. The result is similar to (7) in the single user case

$$\Pr(b_n = b \mid \mathbf{H}, \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{H}, b_n = b)\Pr(b_n = b)}{p(\mathbf{y})} \tag{11}$$

The right-hand side of (11) is obtained by marginalizing over other symbols as

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{H}, b_n = b)\Pr(b_n = b) &= p(\mathbf{y}, b_n = b \mid \mathbf{H}) \\ &= \sum_{\mathbf{b} \in b_n = b} p(\mathbf{y}, \mathbf{b} \mid \mathbf{H}) \\ &= \sum_{\mathbf{b} \in b_n = b} p(\mathbf{y} \mid \mathbf{H}, \mathbf{b})\Pr(\mathbf{b}) \end{aligned} \tag{12}$$

$p(\mathbf{y})$ in (11) is simply a normalization term, and can be expressed as

$$p(\mathbf{y}) = \sum_{\mathbf{b}} p(\mathbf{y}, \mathbf{b} \mid \mathbf{H}) = \sum_{\mathbf{b}} p(\mathbf{y} \mid \mathbf{H}, \mathbf{b})\Pr(\mathbf{b}) \tag{13}$$

Finally, combining (11), (12), and (13), the MAP probability becomes

$$\Pr(b_n = b \mid \mathbf{H}, \mathbf{y}) = \frac{\sum_{\mathbf{b} \in b_n = b} p(\mathbf{y} \mid \mathbf{H}, \mathbf{b})\Pr(\mathbf{b})}{\sum_{\mathbf{b}} p(\mathbf{y} \mid \mathbf{H}, \mathbf{b})\Pr(\mathbf{b})} \tag{14}$$

which is easily calculated given the *a priori* probabilities $\Pr(\mathbf{b})$ for all users and the likelihood values $p(\mathbf{y} \mid \mathbf{H}, \mathbf{b})$ from (8). To finish the technique, the decision is made on this new conditional probability.

$$\hat{b}_n = \arg\max_{b} \Pr(b_n = b \mid \mathbf{H}, \mathbf{y}) \tag{15}$$

The log form of the probabilities is appropriate for this technique and is used in the simulations of chapter 3 and 4. The accuracy of some of the joint probabilities is severely reduced in Matlab due to rounding errors. Any values that are below $10^{-47}$ are

approximated as zero, resulting in error floors. The log form can circumvent this problem. In log form, (14) becomes

$$log\left(Pr\left(b_n = b \,|\, \mathbf{H}, \mathbf{y}\right)\right) = log\left(\frac{\sum_{\mathbf{b} \in b_n = b} e^{ln(p(\mathbf{y}|\mathbf{H},\mathbf{b})) + ln(Pr(\mathbf{b}))}}{\sum_{\mathbf{b}} e^{ln(p(\mathbf{y}|\mathbf{H},\mathbf{b})) + ln(Pr(\mathbf{b}))}}\right) \tag{16}$$

where all the *a priori* probabilities are log form and the conditional likelihood is from (8). The approximation problem in Matlab occurs in the sum of the exponentials. With log-form, the marginalization is done without taking the exponential of individual probabilities. This is done through with the use of the Jacobian algorithm, which also allows for complexity reduction. The reduction is in the form of a tabular approximation, explained below. This technique is known as log-MAP [5]. The Jacobian algorithm states that for log probabilities $\delta$

$$log\left(e^{\delta_1} + e^{\delta_2}\right) = max\left(\delta_1, \delta_2\right) + log\left(1 + e^{-|\delta_2 - \delta_1|}\right) \tag{17}$$

The first term, the maximization, is very simple and does not require any exponential function. The second term depends on the difference of the two terms; thus, if one of the terms is vanishingly small, the larger term will dominate and may prevent an approximation. It has been found that the second term can be approximated as a table of around 10 values without degrading the operation. Therefore, no exponential functions need to be evaluated in this algorithm. For more than two terms, the Jacobian is easily extended

$$\begin{aligned} &log\left(e^{\delta_1} + e^{\delta_2} + \cdots + e^{\delta_n}\right) = \\ &max\left(\delta_1, log\left(e^{\delta_2} + \cdots + e^{\delta_n}\right)\right) + log\left(1 + e^{-\left|log\left(e^{\delta_2} + \cdots + e^{\delta_n}\right) - \delta_1\right|}\right) \end{aligned} \tag{18}$$

For the simulations, the use of a table instead of the second term was not investigated; the log and exponential functions were evaluated. The algorithm did remove the approximation problems in Matlab. For applications with limited computational power, it appears to be very straightforward to apply the tabular approximation

### 2.4.3 Zero-Forcing

Some of the original work in MUD was on linear techniques due to their simplicity. Decorrelation MUD, more commonly known as zero forcing (ZF), is the most straightforward of the linear techniques. Wolniansky et al. discuss ZF MUD and its application to the V-BLAST MUD covered in section 2.4.5.

Zero forcing is essentially a well-known beam forming technique using the pseudo-inverse (detailed thoroughly in [7]). With knowledge of the channel state for all users, the zero-forcing MUD makes a decision on the $j^{th}$ user by nulling out the $N$-1 interferers. **W**, the zero-forcing filter, is the pseudo-inverse of **H**. The reason this solution works is demonstrated in the following simple derivation, where we start with (2)

$$\begin{aligned} \mathbf{y} &= \mathbf{H}\mathbf{b} + \mathbf{n} \\ \mathbf{H}^{\dagger}\mathbf{y} &= \mathbf{H}^{\dagger}\mathbf{H}\mathbf{b} + \mathbf{H}^{\dagger}\mathbf{n} \\ \left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1}\mathbf{H}^{\dagger}\mathbf{y} &= \left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1}\left(\mathbf{H}^{\dagger}\mathbf{H}\right)\mathbf{b} + \left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1}\mathbf{H}^{\dagger}\mathbf{n} \\ \mathbf{W}^{\dagger}\mathbf{y} &= \mathbf{b} + \mathbf{W}^{\dagger}\mathbf{n} \\ \text{so } \mathbf{W} &= \mathbf{H}\left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1} \end{aligned} \tag{19}$$

To use the filter to solve for the users, we simply take the product of the conjugate transpose of **W** and **y**

$$\mathbf{W}^{\dagger}\mathbf{y} = \mathbf{b} + \left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1}\mathbf{H}^{\dagger}\mathbf{n} \tag{20}$$

which is the transmitted user symbols plus a filtered noise. For BPSK, we can then take the sign of (20) as our symbol estimate

$$\hat{\mathbf{b}} = \text{sign}\left(\mathbf{W}^{\dagger}\mathbf{y}\right) \tag{21}$$

Although the linear techniques are simple, they have certain drawbacks. ZF forces contributions from all users except the current user under detection to zero (hence the term zero-forcing). If the users are collinear or have linearly dependent gain vectors, the nulling of certain unwanted users will reduce the SNR of the desired user. This is a problem in all linear techniques and referred to as noise enhancement. Noise enhancement is reduced in the next linear technique, minimum mean square error MUD.

ZF MUD can achieve decent results in systems where there are at least as many receive antennas as users. These are termed overdetermined systems, where $N \leq M$. In this case, ZF MUD exhibits diversity order equivalent to $(M-N)+1$. If there are equal receive antennas and users, the BER curve with a log-log scale will have a slope of 1, or a decade reduction in BER per 10dB increase of SNR due to the diversity. For every extra receive antenna added, an order of diversity will be gained. In overloaded cases (also called underdetermined) where $N < M$, ZF falls short since it does not have the ability to completely null out all interferers. As seen in the overdetermined case, it takes one antenna to null out one user. After all users have been nulled, the remaining antennas can be used to add diversity to the system. For the overloaded case, all users cannot be nulled; the matrix $\mathbf{H}^{\dagger}\mathbf{H}$ is singular and not invertible.

### 2.4.4 Minimum Mean Square Error

In [6], Winters examines the minimum mean square error (MMSE) technique with users of equal and varying power levels. In our analysis, we have equi-power users, all of which are desired. MMSE is the most practical linear technique devised. It reduces

the noise enhancement problem encountered in ZF MUD by including the noise power in the filter calculations. At the least, this avoids the singularity encountered by the ZF filter in overloaded cases.

MMSE is based on the classic error minimization calculation, shown commonly as the Weiner-Hopf equations. From [7], we see the Weiner-Hopf equation as the basis of the MMSE filter, expressed as

$$\mathrm{E}\left[\mathbf{y}\mathbf{y}^{\dagger} \mid \mathbf{H}\right]\mathbf{W} = \mathrm{E}\left[\mathbf{y}\mathbf{b}^{\dagger} \mid \mathbf{H}\right] \tag{22}$$

where $\mathbf{W}$ is our $M \times N$ MMSE filter matrix, $\mathbf{y}$ is our received samples and $\mathbf{H}$ is our channel state matrix. Solving for the expected values, we get

$$\begin{aligned} \left(\mathbf{H}\mathbf{H}^{\dagger} + N_0\mathbf{I}\right)\mathbf{W} &= \mathbf{H} \\ \mathbf{W} &= \mathbf{R}_{yy}^{-1}\mathbf{H} \end{aligned} \tag{23}$$

where

$$\mathbf{R}_{yy} = \mathbf{H}\mathbf{H}^{\dagger} + N_0\mathbf{I} \tag{24}$$

The inclusion of the noise power is evident in (23), with a power of $N_0$.

As mentioned above, the SNR knowledge allows MMSE MUD to minimize the sum of the noise and the interference variances, so it doesn't focus on either variance exclusively. Therefore, MMSE MUD does not try to null out the smallest interferers; it prevents noise enhancement by scaling the matrix inverse. In ZF, the desired user gain is normalized to 1. This normalization can also enhance the noise. The inclusion of the noise power in MMSE acts as a weight; if the noise is very large compared to the channel information in (23), it reduces $\mathbf{W}$ proportionately. This removes the normalization of the desired user, but keeps the SNR from being further reduced. The effect is evident as a shift of the BER curve.

MMSE MUD has the same diversity effect as ZF MUD; it performs poorly in overloaded systems, and has an asymptotic diversity order of $(M-N)+1$ when $N \geq M$.

### 2.4.5 Vertical BLAST

Vertical BLAST (Bell Labs Space-Time) was first examined in [8*]. Usually referred to as V-BLAST, it is essentially an extension of the linear methods detailed above. V-BLAST introduces two techniques: decision feedback and channel dependent decision ordering.

In ZF and MMSE MUD, there is no order to the users detection; all users are detected simultaneously through the filter matrix **W**. The idea behind V-BLAST is to use existing knowledge about the reliability of a users decision to improve the MUD process. The users are detected in a specific order, with the 'best' user detected individually and then effectively removed from the received samples with hard interference cancellation (IC). The 'hard' in hard IC refers to the fact that a symbol decision is made and then used to remove that user's contribution to the vector **y**. The criterion for 'best' is up for debate, and depends on performance and complexity.

[8] used ZF MUD and the SNR as a way of determining the best user. Wolniansky et al. first ranked all the users based on their SNR. First, the zero-forcing filters for all users is calculated. The SNR for all users is determined with the filter. Next, a hard decision is found with the filter for the user with the highest calculated SNR. The decision is then removed from **y**. For the $j^{th}$ user, the modified received samples becomes

$$\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} - \mathbf{h}_j \hat{b}_j \tag{25}$$

where $\mathbf{y}^{(j)}$ is the received samples used in detection of the $j^{th}$ user, $\mathbf{h}_j$ is the channel gain vector for the $j^{th}$ user, and $\hat{b}_j$ is the hard decision made for the $j^{th}$ user. The remaining users are re-evaluated to determine which, in the absence of the previous user(s), has the highest SNR. The filter is recalculated, and the SNR's compared. The process is repeated until all users are detected. Other ordering strategies include signal to interference and noise ratio (SINR), reliability [9], log-likelihood ratio [10] and error variance.

The main complexity in V-BLAST is focused in the ordering technique. The ordering technique can be relatively complex due to matrix inverses. However, if the channel is quasi-static (the channel states are the same for a number of symbol intervals) or static, then the ordering only needs to be accomplished at the beginning of the current static block. This lowers the complexity of V-BLAST to something comparable to ZF or MMSE MUD; the difference is in the decision cancellation.

### 2.4.6 Group Detection

Group detection is a combination of the optimum ML or MAP technique with that of a linear technique. The term 'group' comes from the fact that all $N$ users are separated into $N_G$ groups, each with $G$ users. A group is detected by suppressing all the other $N_G - 1$ groups with a linear technique. Then, the JML or JMAP is applied to the group users with the filtered received samples.

Fain and Varanasi first examined group detection in a narrow-band system with a ZF filter for group suppression in [11]. Fain and Varanasi noted that group detection spans the performance and complexity space between JML and ZF MUD. If $N_G = N$, ZF MUD results; conversely, if $N_G = 1$, the group detection becomes JML MUD. Increasing

the number of groups in group detection decreases the performance and increases the complexity, since it is more complex to calculate $2^N$ metrics than it is $N_G 2^G$. For example, for 6 users with 3 groups of 2 users, JML needs to calculate $2^6 = 64$ metrics while group detection only calculates $3 \cdot 2^2 = 12$ metrics.

The formulation of the ZF group detector is straightforward. The ZF filter for all $N$ users is calculated from (19). The $M \times N$ $\mathbf{W}$ matrix contains a filter for all $N$ users. The filter matrix that contains the filters for the $j^{th}$ group of users is denoted as $\mathbf{W}_j$. The filtered received sample is simply

$$\mathbf{x} = \mathbf{W}^\dagger \mathbf{y} = \begin{bmatrix} \mathbf{x}_j \\ \mathbf{x}_{\bar{j}} \end{bmatrix} \tag{26}$$

$\mathbf{x}_j$ is a $G \times 1$ column vector that is formed from the users in the $j^{th}$ group of $\mathbf{x}$, and $\mathbf{x}_{\bar{j}}$ consists of all other users. Since the received samples are changed, so are some of the assumptions made during the formulation of the JML. The joint ML decisions are now made on $\mathbf{x}_j$ instead of $\mathbf{y}$. The mean in (8) changes to

$$\begin{aligned} \mathrm{E}[\mathbf{x} \mid \mathbf{H}, \mathbf{b}] &= \mathrm{E}\left[\mathbf{W}^\dagger \mathbf{y} \mid \mathbf{H}, \mathbf{b}\right] \\ &= \mathbf{W}^\dagger \mathrm{E}[\mathbf{y} \mid \mathbf{H}, \mathbf{b}] \\ &= \left(\mathbf{H}^\dagger \mathbf{H}\right) \mathbf{H}^\dagger \mathbf{H} \mathbf{b} \\ &= \mathbf{b} \end{aligned} \tag{27}$$

Segmenting $\mathbf{H}$ into sub-matrices for both the users in group $j$ and for all other users, $\mathbf{H} = \left[\mathbf{H}_j \mid \mathbf{H}_{\bar{j}}\right]$, the variance for the ZF group detector becomes

$$\begin{aligned}\mathbf{Q} &= \mathrm{E}\left[\left(\mathbf{x}-\mathrm{E}[\mathbf{x}\,|\mathbf{H},\mathbf{b}]\right)\left(\mathbf{x}-\mathrm{E}[\mathbf{x}\,|\mathbf{H},\mathbf{b}]\right)^{\dagger}\,|\mathbf{H},\mathbf{b}\right] \\ &= \mathrm{E}\left[\left(\left(\mathbf{b}-\left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1}\mathbf{H}^{\dagger}\mathbf{n}\right)-\mathbf{b}\right)\left(\left(\mathbf{b}-\left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1}\mathbf{H}^{\dagger}\mathbf{n}\right)-\mathbf{b}\right)^{\dagger}\,|\mathbf{H},\mathbf{b}\right] \\ &= N_0\left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1} = N_0\begin{pmatrix}\mathbf{H}_j^{\dagger}\mathbf{H}_j & \mathbf{H}_j^{\dagger}\mathbf{H}_{\bar{j}} \\ \mathbf{H}_{\bar{j}}^{\dagger}\mathbf{H}_j & \mathbf{H}_{\bar{j}}^{\dagger}\mathbf{H}_{\bar{j}}\end{pmatrix}^{-1} = \begin{pmatrix}\mathbf{Q}_j & \mathbf{Q}_{j\bar{j}} \\ \mathbf{Q}_{\bar{j}j} & \mathbf{Q}_{\bar{j}}\end{pmatrix}\end{aligned} \tag{28}$$

So, for ZF group detection the JML metric becomes

$$\hat{\mathbf{b}}_j = \underset{\mathbf{b}_j}{arg\,min}\left(\mathbf{x}_j-\mathbf{b}_j\right)^{\dagger}\mathbf{Q}_j^{-1}\left(\mathbf{x}_j-\mathbf{b}_j\right) \tag{29}$$

for each group.

In [9], Ng and Sousa replaced the ZF filter with an MMSE filter. For the MMSE filter, $\mathbf{x}_j$, $\mathbf{W}_j$ and $\mathbf{Q}_j$ for the $j^{th}$ group can be formed independent of the other groups. $\mathbf{W}_j$ and $\mathbf{x}_j$ become

$$\begin{aligned}\mathbf{W}_j &= \mathbf{R}_{yy}^{-1}\mathbf{H}_j \\ &= \left(\sigma_b^2\mathbf{H}\mathbf{H}^{\dagger}+N_0\mathbf{I}_M\right)^{-1}\mathbf{H}_j \\ \mathbf{x}_j &= \mathbf{W}_j^{\dagger}\mathbf{y}\end{aligned} \tag{30}$$

where $\mathbf{H}_j$ is an $M\times G$ matrix of the channel gains from the $j^{th}$ group of $G$ users to all $M$ antennas. $\sigma_b^2$ is the variance of the transmitted symbols, and for BPSK is equal to 1.

$\mathbf{Q}_j$ becomes

$$
\begin{aligned}
\mathbf{Q}_j &= \mathrm{E}\left[\left(\mathbf{x}_j - \mathrm{E}\left[\mathbf{x}_j \mid \mathbf{H}, \mathbf{b}_j\right]\right)\left(\mathbf{x}_j - \mathrm{E}\left[\mathbf{x}_j \mid \mathbf{H}, \mathbf{b}_j\right]\right)^{\dagger} \mid \mathbf{H}, \mathbf{b}_j\right] \\
&= \mathrm{E}\left[\left(\left(\mathbf{W}_j^{\dagger}\mathbf{H}\mathbf{b} + \mathbf{W}_j^{\dagger}\mathbf{n}\right) - \mathbf{W}_j^{\dagger}\mathbf{H}_j\mathbf{b}_j\right)\left(\left(\mathbf{W}_j^{\dagger}\mathbf{H}\mathbf{b} + \mathbf{W}_j^{\dagger}\mathbf{n}\right) - \mathbf{W}_j^{\dagger}\mathbf{H}_j\mathbf{b}_j\right)^{\dagger} \mid \mathbf{H}, \mathbf{b}_j\right] \\
&= \mathbf{W}_j^{\dagger}\mathrm{E}\left[\left(\sum_{\substack{i=1 \\ i \neq j}}^{N_G}\mathbf{H}_i\mathbf{b}_i + \mathbf{n}\right)\left(\sum_{\substack{i=1 \\ i \neq j}}^{N_G}\mathbf{H}_i\mathbf{b}_i + \mathbf{n}\right)^{\dagger} \mid \mathbf{H}, \mathbf{b}_j\right]\mathbf{W}_j \\
&= \mathbf{W}_j^{\dagger}\left(\sum_{\substack{i=1 \\ i \neq j}}^{N_G}\mathbf{H}_i\mathrm{E}\left[\mathbf{b}_i\mathbf{b}_i^{\dagger}\right]\mathbf{H}_i^{\dagger} + N_0\mathbf{I}\right)\mathbf{W}_j \\
&= \mathbf{W}_j^{\dagger}\left(\sum_{\substack{i=1 \\ i \neq j}}^{N_G}\mathbf{H}_i\mathbf{H}_i^{\dagger} + N_0\mathbf{I}\right)\mathbf{W}_j
\end{aligned}
\tag{31}
$$

since the transmitted symbols are uncorrelated with unit variance. The metric for MMSE group detection becomes

$$
\hat{\mathbf{b}}_j = \underset{\mathbf{b}_j}{arg\,min}\left(\mathbf{x}_j - \mathbf{W}_j^{\dagger}\mathbf{H}_j\mathbf{b}_j\right)^{\dagger}\mathbf{Q}_j^{-1}\left(\mathbf{x}_j - \mathbf{W}_j^{\dagger}\mathbf{H}_j\mathbf{b}_j\right) \tag{32}
$$

Ng and Sousa [9] also investigated the use of interference cancellation between groups. Ng and Sousa used a metric termed the 'reliability metric' to order the groups with respect to one another. The metric is formed by summing the JML metrics (8) for all possible user symbol combinations. Each group's reliability, which is the sum of the JML metrics, is then compared. The group with the lowest sum is detected first. This is unlike V-BLAST, which orders the users individually. Once a group is detected, the hard decisions are used to remove the group's contribution to the received samples. After detection of the $j^{th}$ group, the modified received samples become

$$
\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} - \mathbf{H}_j\hat{\mathbf{b}}_j \tag{33}
$$

where $\mathbf{y}^{(j)}$ is the received samples used in detection of the $j^{th}$ group, $\mathbf{H}_j$ is the channel matrix for the users in the $j^{th}$ group, and $\hat{\mathbf{b}}_j$ is the hard decisions made for the $j^{th}$ group.

The Ng and Sousa technique, which uses GD with interference cancellation between groups and the reliability metric for ordering, observed gains of a few dB over MMSE MUD. From this point on, any reference to group detection, or GD, will be made to the MMSE and hard interference cancellation version of Ng and Sousa.

## 2.5 Iterative Detection Techniques

Iterative techniques have become quite popular in recent years due to the advent of Turbo codes. Turbo codes are actually conventional convolutional codes used in a novel, iterative manner. They have brought communications system tantalizingly close to the Shannon limit, the theoretical capacity limit of communications system. Turbo codes achieve these amazing feats through an iterative technique involving the transfer of symbol probabilities from iteration to iteration. The idea is simple in its basic form, and is covered in detail in [5]. A summary is as follows: two convolutional codes are used at the encoder. One encodes the original data stream, while the second encodes an interleaved version. Both streams are multiplexed and transmitted. The encoder is shown in Figure 7. The receiver starts detection by de-multiplexing the streams. The first stream is decoded on the appropriate code trellis using the BJCR algorithm [12]. The BCJR outputs *a posteriori* probabilities (APP). However, the only information forwarded to the next stage of the decoder is that termed extrinsic information; extrinsic information is a product of the redundancy of the convolutional codes used. These extrinsic APPs are interleaved in the same manner as at the transmitter. They are then used in the second code trellis with the second received coded sequence as *a priori* information. Again, the second BCJR algorithm outputs extrinsic APPs, which are then de-interleaved. That ends the first iteration of the Turbo code; the structure for the decoder is shown in Figure 8 below. For the second iteration, the extrinsic APPs from the second decoder are used in the first decoder as *a prioris*. This process continues

until a decision is desired, where the decision is made on the de-interleaved output APPs from the second decoder.

**Figure 7: Turbo encoder**

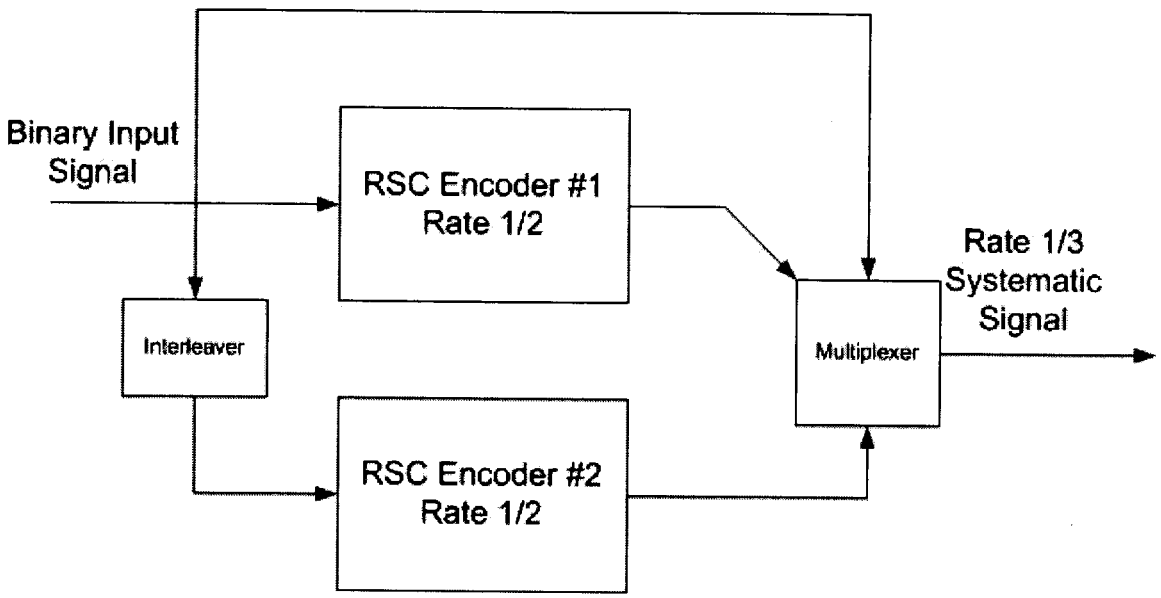
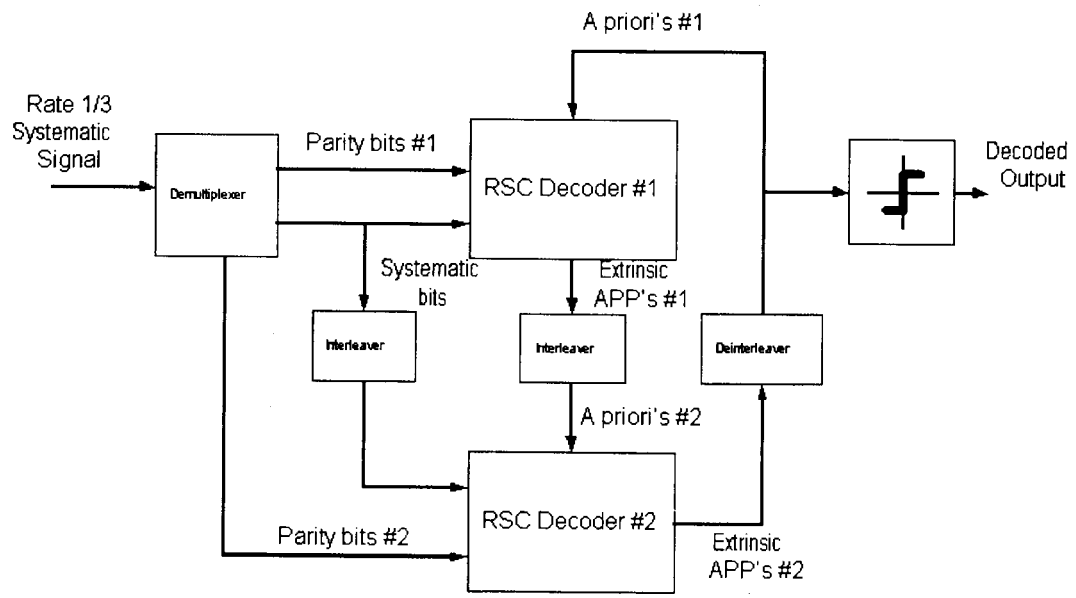


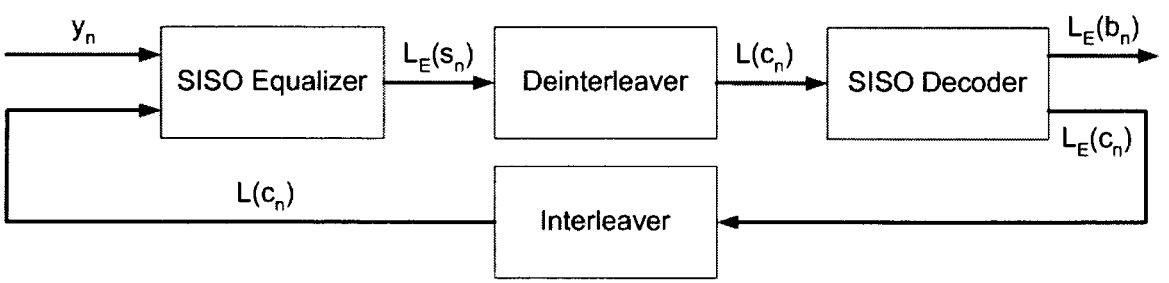

**Figure 8: Turbo decoder**



The idea of transferring APPs in an iterative fashion has been successfully used in other applications than Turbo codes. Turbo codes are parallel concatenated codes, where each code works on the same data, only in a different order. Serially concatenated codes have also been decoded in an iterative fashion with good results. Benedetto and Montorsi [13] showed that the serially concatenated codes can actually outperform the parallel concatenated Turbo codes.

An extension of both Turbo and serially concatenated codes is Turbo equalization, where the second code is replaced by the multi-path channel. Tuchler et al. review current techniques for Turbo equalization in [14]. The multipath channel is used as a channel dependent "code" with its own trellis. The data stream from the transmitter can be viewed as being serially encoded with the channel. The structure of the receiver is shown in Figure 9 below.

**Figure 9: Turbo equalizer decoder**




The SISO equalizer acts as the outer decoder, using the multipath for detection. Next, any interleaving done at the transmitter to avoid error clustering is de-interleaved. A SISO decoder (covered in Section 4) uses the information from the soft equalizer to decode the transmitter's convolutional code.

The information passed between the blocks in all of these iterative techniques are referred to as log-likelihood ratios (LLR) and are equivalent to the symbol

probabilities. For simplicity, only BPSK systems are covered here. The LLR of a symbol $b$ is denoted

$$L(b) = ln\left(\frac{Pr(b=+1)}{Pr(b+-1)}\right) \tag{34}$$

and the LLR of a symbol $b$ conditioned on a received sample $y$ is denoted

$$L(b|y) = ln\left(\frac{Pr(b=+1|y)}{Pr(b=-1|y)}\right) \tag{35}$$

In Figure 9 above, $L_E(s_n)$ refers to the extrinsic LLR of $s_n$, the coded and interleaved $c_n$ symbols. The concept of extrinsic LLR is examined in detail in Section 3.8.5. After the de-interleaver, this LLR becomes $L(c_n)$, which refers to the LLR of the coded symbols in their proper order.

Finally, exploration of the combination of iterative and MUD techniques is also underway. A recent article by Poor [15] reviews the current state of the art. The structure that Poor presents is very similar to the Turbo equalizer above, where the multiuser detector is used as an inner decoder instead of the convolutional code. The MUD is converted to allow for soft-input/soft-output. Its output is then de-interleaved and each user's detected symbols are fed to their respective SISO decoder.

# 3 ITERATIVE MULTIUSER DETECTION

The focus of this thesis is the investigation of a new multiuser detection technique, referred to as iterative multiuser detection or IMUD. IMUD is an extension of the groupwise detector, using interference cancellation, user ordering, and iterative techniques. These techniques were introduced in chapter 2.

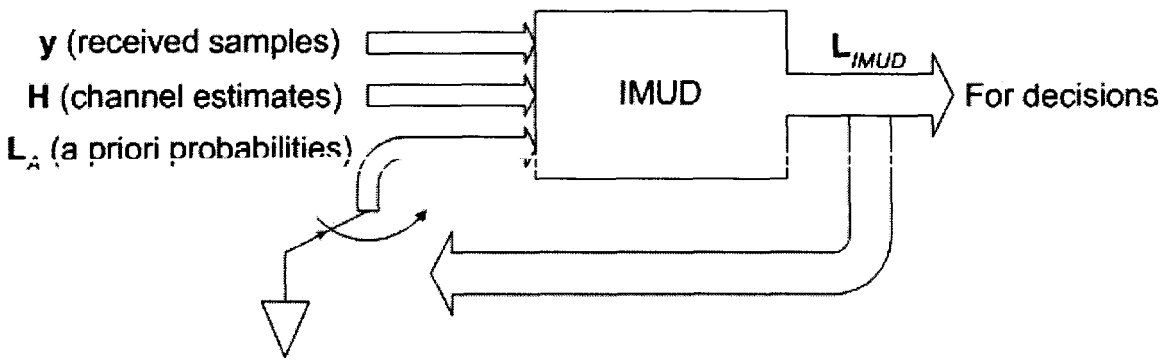
The novel features of IMUD include user interleaving and soft decisions between groups and iterations. Most iterative techniques interleave symbols in time, since the memory in the system is achieved through using two separate encoders or one encoder and a multipath channel in the case of turbo equalization; the interleaving between the memory components breaks any correlations that exist between the APPs of each component, allowing the next to work on uncorrelated symbols. The transference of the *a posteriori* probabilities (APPs) between the components allows the information gained in the current component and iteration to be used in the next. In IMUD, group detection separates the users into groups, which are each then detected in succession with the aid of linear suppression of the remaining users. Since IMUD is a symbol-by-symbol detector, there is no time dependence, thus a time oriented interleaver is useless. However, the APP extraction of each group adds correlation to the group's soft decision. Through the use of random interleaving of users between iterations, the information gained from the past iteration will experience a different correlation in the next. The information gained from each iteration of different user combinations allows the technique to achieve a better performance than a group detector.

IMUD will first be introduced as a groupwise APP extraction block, shown in Figure 10. This block is not iterative itself, but it will be used to show how the iterative

detection technique is formed. For the first iteration, IMUD is equivalent to GD using joint APP extraction (JMAP without a decision) with soft interference cancellation and EVM user ordering. For subsequent iterations, IMUD uses the probabilities from the last iteration as *a prioris* and the groups are randomly reformed.

The inputs necessary to use IMUD are similar to the previous multiuser detectors. Since IMUD is SISO, it also makes use of any *a priori* information available. This allows IMUD to be incorporated easily into a serially concatenated scheme similar to Turbo equalization or Turbo MUD from chapter 2. In log-likelihood form, the outputs of IMUD correspond to the soft-output MUD in [15], and will be referred to as $\mathbf{L}_{IMUD}$.

**Figure 10: IMUD detection block**




Use of IMUD as an iterative system is straightforward. $\mathbf{L}_A$ are the *a priori* probabilities provided by the other iterative component in the detector; in IMUD's case, this is either the previous iteration of IMUD or another SISO block, such as a convolutional decoder. The output APPs, $\mathbf{L}_{IMUD}$, is used as *a prioris* in the next step. Issues regarding the feedback of the APPs are detailed near the end of this chapter. Further detail for concatenated systems is given in chapter 4. As mentioned briefly in chapter 2, $\mathbf{L}_{IMUD}(\mathbf{b})$ is the log-likelihood ratio of all symbols in **b**. Each component is calculated as

$$\mathbf{L}_{IMUD}(b_i) = log\left(\frac{Pr(b_i = +1 \mid \mathbf{y})}{Pr(b_i = -1 \mid \mathbf{y})}\right) \tag{36}$$

## 3.1 Formulation of IMUD Groups

The IMUD group detection is MMSE GD with joint APP extraction on each group. Joint APP extraction is simply JMAP without making a decision on the output probabilities. The soft-output is simply the probabilities found in (14).

The MMSE suppression is altered to an equivalent method that has the potential to be computationally less expensive. An equivalent to group detection with MMSE filter suppression is a group joint technique with the assumption that all undesired users are Gaussian noise. We call this technique group JML (GML) or group APP extraction (GAPP) for hard and soft outputs, respectively. In the following, we will demonstrate the equivalence with a simple derivation for GML. First, we reform (2) showing the current decision or desired group, group *j*, and all undesired groups.

$$\begin{aligned}\mathbf{y} &= \left(\mathbf{H}_j\mathbf{b}_j\right) + \left(\sum_{\substack{i=1 \\ i\neq j}}^{N_G}\mathbf{H}_i\mathbf{b}_i + \mathbf{n}\right) \\ &= \mathbf{y}_D + \mathbf{y}_U\end{aligned} \tag{37}$$

where $\mathbf{y}_D$ refers to $\mathbf{H}_j\mathbf{b}_j$, the current desired decision group, and $\mathbf{y}_U$ refers to the sum of all other groups and the receiver noise. If we use the assumption that $\mathbf{y}_U$ is approximately Gaussian, we can fit this to a likelihood function $p\left(\mathbf{y} \mid \mathbf{H}, \mathbf{b}_j\right)$ just as in (8). Now we are looking only for $\mathbf{b}_j$ instead of the entire user set and our noise has the undesired users added. The mean of $\mathbf{y}$ in (37) for the $j^{th}$ group is

$$E\left[\mathbf{y} \mid \mathbf{H}, \mathbf{b}_j\right] = \mathbf{H}_j\mathbf{b}_j \tag{38}$$

The covariance of **y** is

$$
\begin{aligned}
\mathbf{R}_U &= \mathrm{E}\left[\left(\mathbf{y}-\mathbf{H}_j\mathbf{b}_j\right)\left(\mathbf{y}-\mathbf{H}_j\mathbf{b}_j\right)^{\dagger} \,|\, \mathbf{H},\mathbf{b}_j\right] \\
&= \mathrm{E}\left[\left(\sum_{\substack{i=1\\ i\neq j}}^{N_G}\mathbf{H}_i\mathbf{b}_i+\mathbf{n}\right)\left(\sum_{\substack{i=1\\ i\neq j}}^{N_G}\mathbf{H}_i\mathbf{b}_i+\mathbf{n}\right)^{\dagger} \,|\, \mathbf{H},\mathbf{b}_j\right] \\
&= \mathrm{E}\left[\sum_{\substack{i=1\\ i\neq j}}^{N_G}\mathbf{H}_i\mathbf{b}_i\mathbf{b}_i^{\dagger}\mathbf{H}_i^{\dagger}+\mathbf{n}\sum_{\substack{i=1\\ i\neq j}}^{N_G}\mathbf{b}_i^{\dagger}\mathbf{H}_i^{\dagger}+\sum_{\substack{i=1\\ i\neq j}}^{N_G}\mathbf{H}_i\mathbf{b}_i\mathbf{n}^{\dagger}+\mathbf{n}\mathbf{n}^{\dagger} \,|\, \mathbf{H},\mathbf{b}_j\right] \\
&= \sum_{\substack{i=1\\ i\neq j}}^{N_G}\mathbf{H}_i\mathrm{E}\left[\mathbf{b}_i\mathbf{b}_i^{\dagger}\right]\mathbf{H}_i^{\dagger}+\mathrm{E}[\mathbf{n}]\sum_{\substack{i=1\\ i\neq j}}^{N_G}\mathrm{E}\left[\mathbf{b}_i^{\dagger}\right]\mathbf{H}_i^{\dagger}+\sum_{\substack{i=1\\ i\neq j}}^{N_G}\mathbf{H}_i\mathrm{E}[\mathbf{b}_i]\mathrm{E}\left[\mathbf{n}^{\dagger}\right]+\mathrm{E}\left[\mathbf{n}\mathbf{n}^{\dagger}\right] \\
&= \sum_{\substack{i=1\\ i\neq j}}^{N_G}\mathbf{H}_i\mathbf{H}_i^{\dagger}+N_0\mathbf{I}
\end{aligned}
\tag{39}
$$

since the users are uncorrelated with each other and with the noise. In (8), it was assumed that $\mathbf{R}_U$ was simply a diagonal matrix since the only undesired quantity was the Gaussian noise and our users are uncorrelated. (39) shows how it is extended to include other users as Gaussian interferers.

Putting (38) and (39) together, we show the likelihood function as

$$
p\left(\mathbf{y}\,|\,\mathbf{H},\mathbf{b}_j\right)=\frac{1}{2\pi|\mathbf{R}_U|}e^{-\frac{1}{2}\left(\mathbf{y}-\mathbf{H}_j\mathbf{b}_j\right)^{\dagger}\mathbf{R}_U^{-1}\left(\mathbf{y}-\mathbf{H}_j\mathbf{b}_j\right)} \tag{40}
$$

(40) can be reduced to the log-form metric and decision method

$$
\hat{\mathbf{b}}_j=\underset{\mathbf{b}_j}{arg\,min}\left(\mathbf{y}-\mathbf{H}_j\mathbf{b}_j\right)^{\dagger}\mathbf{R}_U^{-1}\left(\mathbf{y}-\mathbf{H}_j\mathbf{b}_j\right) \tag{41}
$$

To show the equivalence between GML and the MMSE GD in [9], the following derivation is presented.

$$
\begin{aligned}
\hat{\mathbf{b}}_j &= \arg\min_{\mathbf{b}_j} \left(\mathbf{x}_j - \mathbf{W}_j^{\dagger}\mathbf{H}_j\mathbf{b}_j\right)^{\dagger} \mathbf{Q}_j^{-1} \left(\mathbf{x}_j - \mathbf{W}_j^{\dagger}\mathbf{H}_j\mathbf{b}_j\right) \\
&= \arg\min_{\mathbf{b}_j} \left(\mathbf{W}_j^{\dagger}\mathbf{y} - \mathbf{W}_j^{\dagger}\mathbf{H}_j\mathbf{b}_j\right)^{\dagger} \mathbf{Q}_j^{-1} \left(\mathbf{W}_j^{\dagger}\mathbf{y} - \mathbf{W}_j^{\dagger}\mathbf{H}_j\mathbf{b}_j\right) \\
&= \arg\min_{\mathbf{b}_j} \left(\left(\mathbf{y} - \mathbf{H}_j\mathbf{b}_j\right)^{\dagger} \mathbf{W}_j\right) \mathbf{Q}_j^{-1} \left(\mathbf{W}_j^{\dagger}\left(\mathbf{y} - \mathbf{H}_j\mathbf{b}_j\right)\right) \\
&= \arg\min_{\mathbf{b}_j} \left(\left(\mathbf{y} - \mathbf{H}_j\mathbf{b}_j\right)^{\dagger} \mathbf{W}_j\right) \left(\mathbf{W}_j^{\dagger}\left(\mathbf{R}_U\right)\mathbf{W}_j\right)^{-1} \left(\mathbf{W}_j^{\dagger}\left(\mathbf{y} - \mathbf{H}_j\mathbf{b}_j\right)\right) \\
&= \arg\min_{\mathbf{b}_j} \left(\left(\mathbf{y} - \mathbf{H}_j\mathbf{b}_j\right)^{\dagger} \mathbf{W}_j\right) \left(\mathbf{W}_j\right)^{-1} \mathbf{R}_U^{-1} \left(\mathbf{W}_j^{\dagger}\right)^{-1} \left(\mathbf{W}_j^{\dagger}\left(\mathbf{y} - \mathbf{H}_j\mathbf{b}_j\right)\right) \\
&= \arg\min_{\mathbf{b}_j} \left(\mathbf{y} - \mathbf{H}_j\mathbf{b}_j\right)^{\dagger} \mathbf{R}_U^{-1} \left(\mathbf{y} - \mathbf{H}_j\mathbf{b}_j\right)
\end{aligned}
\tag{42}
$$

The simplified decision structure in (42) turns out to be identical to that of GML in (41).

Next, consider the IC component of IMUD. Between groups and iterations, IMUD removes what is known of previously detected symbols from the measurement vector $\mathbf{y}$. For the $j^{\text{th}}$ group and $\gamma^{\text{th}}$ iteration, the modified measurement vector is denoted as $\mathbf{y}^{(j,\gamma)}$ (60) and its covariance matrix is $\mathbf{R}_U^{(j,\gamma)}$ (61). They are derived in Section 3.4 below. The new likelihood function is simply

$$
p\left(\mathbf{y}^{(j,\gamma)} \mid \mathbf{H}, \mathbf{b}_j\right) = \frac{1}{2\pi\left|\mathbf{R}_U^{(j,\gamma)}\right|} e^{-\frac{1}{2}\left(\mathbf{y}^{(j,\gamma)} - \mathbf{H}_j\mathbf{b}_j\right)^{\dagger} \mathbf{R}_U^{(j,\gamma)^{-1}} \left(\mathbf{y}^{(j,\gamma)} - \mathbf{H}_j\mathbf{b}_j\right)} \tag{43}
$$

changing the log-likelihood to

$$
\hat{\mathbf{b}}_j = \arg\min_{\mathbf{b}_j} \left(\mathbf{y}^{(j,\gamma)} - \mathbf{H}_j\mathbf{b}_j\right)^{\dagger} \mathbf{R}_U^{(j,\gamma)^{-1}} \left(\mathbf{y}^{(j,\gamma)} - \mathbf{H}_j\mathbf{b}_j\right) \tag{44}
$$

To change the GML into GAPP, the same marginalizations used to form JMAP are used but (8) is replaced with (43). Section 3.2 provides the details.

We now have two formulations of MMSE GD; both GD with an MMSE filter and GML make a joint decision on a group of *G* users while suppressing the other *N-G* users. For IMUD, either technique converted to give soft outputs would be equally

effective. However, as shown in Section 3.7, GML/GAPP is less computationally complex due to the number of inverses needed.

To summarize, the group techniques are tabulated in table 1.

**Table 1: Feature comparison of groupwise techniques**

| Groupwise Technique | Characteristics |
|---|---|
| Group Detection (GD) | - Hard decision<br>- Uses explicit MMSE filter<br>- No *a priori* information |
| Group ML (GML) | - Hard decision<br>- Uses implicit MMSE filter<br>- No *a priori* information |
| Group MAP (GMAP) | - Hard decision<br>- Uses implicit MMSE filter<br>- Uses *a priori* information |
| Group APP Extraction (GAPP) | - Soft decision<br>- Uses implicit MMSE filter<br>- Uses *a priori* information |

## 3.2 Group APP Extraction

The group technique used in IMUD was created to meet two demands: it must be soft-output and it must be computationally inexpensive. The first criterion is met with a soft-output version of the GD shown in Section 2.4.6. However, as shown in Section 3.1, an equivalent method with less computational complexity (Section 3.7) is realized with the group APP extraction technique.

$\Pr(b_n = k)$ is the *a priori* probability for the $n^{\text{th}}$ user symbol at the beginning of each iteration, where $k$ is $\pm 1$ for BPSK. The goal of the APP extraction is to use the joint probabilities found with (43) and the *a prioris* to find new APPs that will be used for either decisions or to create new LLRs for further iterative processing. The channel knowledge **H** is assumed and left out of the following derivation.

The final APP for the $i^{\text{th}}$ user of the $j^{\text{th}}$ group is denoted by the conditional probability $\Pr\left(b_{ij} \mid \mathbf{y}^{(j,\gamma)}\right)$, where $j$ refers to the $j^{\text{th}}$ group and $\gamma$ refers to the $\gamma^{\text{th}}$ iteration of IMUD. Using the mixed Bayes rule, it can be expressed as

$$\begin{aligned}\Pr\left(b_{ij} = b \mid \mathbf{y}^{(j,\gamma)}\right) &= \frac{\mathrm{p}\left(b_{ij} = b, \mathbf{y}^{(j,\gamma)}\right)}{\mathrm{p}\left(\mathbf{y}^{(j,\gamma)}\right)} \\ &= \frac{\mathrm{p}\left(b_{ij} = b, \mathbf{y}^{(j,\gamma)}\right)}{\sum_{b=\pm 1} \mathrm{p}\left(b_{ij} = b, \mathbf{y}^{(j,\gamma)}\right)}\end{aligned} \tag{45}$$

The joint pdf comprises the conditional pdf or likelihood values (43) and the *a priori* information, as in

$$\mathrm{p}\left(b_{ij} = b, \mathbf{y}^{(j,\gamma)}\right) = \sum_{\mathbf{b}_j \in b_{ij} = b} \left[ \mathrm{p}\left(\mathbf{y}^{(j,\gamma)} \mid \mathbf{b}_j\right) \prod_{k=1}^{G} \Pr\left(b_{kj}\right) \right] \tag{46}$$

where $p\left(\mathbf{y}^{(j,y)} \mid \mathbf{b}_j\right)$ is from (43) and includes IC from previous groups and iterations. $\Pr\left(b_{kj}\right)$ is the *a priori* probability of the $k^{\text{th}}$ user of $\mathbf{b}_j$. The product of the *a priori*'s is the equivalent to $\Pr\left(\mathbf{b}_j\right)$ since the symbols are independent and identically distributed.

## 3.3 User ordering

Before forming groups for GAPP in IMUD, the users are ordered with respect to the error variance minimization (EVM) criterion. The ordering has a significant effect on the performance of IMUD (see the performance comparison in Section 3.8.4). All users are ordered before each group detection takes place, as in V-BLAST [8].

EVM is formulated with the assumption that the decisions are made with MMSE MUD (Section 2.4.4). This is suboptimum, but allows us to develop an analytical and computationally simple expression for the error variance. Minimizing the error has proven to be more effective than strict SNR ordering like that used in [8]. It is shown to be a more reliable metric than other techniques tested (Figure 26, Section 3.8.4).

The error variance minimization technique is as follows. First, all users that have not been detected so far are seen as residing in a 'pile'. The error between the transmitted symbols and the symbols as detected by MMSE MUD is expressed as

$$\mathbf{e} = \mathbf{b} - \hat{\mathbf{b}} \tag{47}$$

where $\hat{\mathbf{b}}$ is found from (21) using the MMSE filter. The preceding vectors only consist of users in the pile; the previously detected users are removed. With the MMSE filter $\mathbf{W}$ defined in (30), the error variance $\mathbf{R}_{\mathbf{ee}}$ of each user in the pile is

$$
\begin{aligned}
\mathbf{R}_{ee} &= \mathrm{E}\left[\mathbf{e}\mathbf{e}^{\dagger}\right] = \mathrm{E}\left[\left(\mathbf{b}-\hat{\mathbf{b}}\right)\left(\mathbf{b}-\hat{\mathbf{b}}\right)^{\dagger}\right] \\
&= \mathrm{E}\left[\mathbf{b}\mathbf{b}^{\dagger} - \mathbf{b}\hat{\mathbf{b}}^{\dagger} - \hat{\mathbf{b}}\mathbf{b}^{\dagger} + \hat{\mathbf{b}}\hat{\mathbf{b}}^{\dagger}\right] \\
&= \mathrm{E}\left[\mathbf{b}\mathbf{b}^{\dagger} - \mathbf{b}\mathbf{y}^{\dagger}\mathbf{W} - \mathbf{W}^{\dagger}\mathbf{y}\mathbf{b}^{\dagger} + \mathbf{W}^{\dagger}\mathbf{y}\mathbf{y}^{\dagger}\mathbf{W}\right] \\
&= \sigma_{\mathrm{b}}^{2}\mathbf{I}_{N} - \mathrm{E}\left[\mathbf{b}\left(\mathbf{H}\mathbf{b}+\mathbf{n}\right)^{\dagger}\mathbf{W} + \mathbf{W}^{\dagger}\left(\mathbf{H}\mathbf{b}+\mathbf{n}\right)\mathbf{b}^{\dagger} - \mathbf{W}^{\dagger}\mathbf{y}\mathbf{y}^{\dagger}\mathbf{W}\right] \\
&= \sigma_{\mathrm{b}}^{2}\mathbf{I}_{N} - \mathrm{E}\left[\mathbf{b}\mathbf{b}^{\dagger}\right]\mathbf{H}^{\dagger}\mathbf{W} - \mathrm{E}\left[\mathbf{b}\mathbf{n}^{\dagger}\right]\mathbf{W} - \mathbf{W}^{\dagger}\mathbf{H}\mathrm{E}\left[\mathbf{b}\mathbf{b}^{\dagger}\right] - \mathbf{W}^{\dagger}\mathrm{E}\left[\mathbf{n}\mathbf{b}^{\dagger}\right] + \mathbf{W}^{\dagger}\mathrm{E}\left[\mathbf{y}\mathbf{y}^{\dagger}\right]\mathbf{W} \\
&= \sigma_{\mathrm{b}}^{2}\mathbf{I}_{N} - \sigma_{\mathrm{b}}^{2}\mathbf{H}^{\dagger}\mathbf{W} - \sigma_{\mathrm{b}}^{2}\mathbf{W}^{\dagger}\mathbf{H} + \mathbf{W}^{\dagger}\mathbf{R}_{\mathbf{yy}}\mathbf{W} \\
&= \sigma_{\mathrm{b}}^{2}\mathbf{I}_{N} - \sigma_{\mathrm{b}}^{4}\mathbf{H}^{\dagger}\mathbf{R}_{\mathbf{yy}}^{-1}\mathbf{H} - \sigma_{\mathrm{b}}^{4}\mathbf{H}^{\dagger}\mathbf{R}_{\mathbf{yy}}^{-1}\mathbf{H} + \sigma_{\mathrm{b}}^{4}\mathbf{H}^{\dagger}\mathbf{R}_{\mathbf{yy}}^{-1}\mathbf{R}_{\mathbf{yy}}\mathbf{R}_{\mathbf{yy}}^{-1}\mathbf{H} \\
&= \sigma_{\mathrm{b}}^{2}\left(\mathbf{I}_{N} - \sigma_{\mathrm{b}}^{2}\mathbf{H}^{\dagger}\mathbf{R}_{\mathbf{yy}}^{-1}\mathbf{H}\right)
\end{aligned}
\tag{48}
$$

where

$$
\begin{aligned}
\mathbf{R}_{\mathbf{yy}} &= \mathrm{E}\left[\mathbf{y}\mathbf{y}^{\dagger}\right] \\
&= \mathrm{E}\left[\left(\mathbf{H}\mathbf{b}+\mathbf{n}\right)\left(\mathbf{H}\mathbf{b}+\mathbf{n}\right)^{\dagger}\right] \\
&= \sigma_{b}^{2}\mathbf{H}\mathbf{H}^{\dagger} + N_{0}\mathbf{I}
\end{aligned}
\tag{49}
$$

and $\sigma_{\mathrm{b}}^{2}$ is the variance of the transmitted symbols. For BPSK, $\sigma_{b}^{2} = 1$ for an undetected user. $\mathbf{R}_{\mathbf{ee}}$ now contains the estimated error variances of all the undetected users. To form the next detection group, the $G$ users with the lowest error variance are used. These $G$ users are removed from the pile and detected with the technique in Section 3.2.

After a group is detected, its gain vectors are removed from the **H** matrix before recalculation of (48). Expressing **H** as a matrix of group channel states $\mathbf{H} = \left[\mathbf{H}_{1} \,|\, \mathbf{H}_{2} \,|\cdots|\, \mathbf{H}_{N_G}\right]$, the channel state for the $j^{\text{th}}$ group is

$$
\mathbf{H}^{(j)} = \left[\mathbf{H}_{j} \,|\, \mathbf{H}_{j+1} \,|\cdots|\, \mathbf{H}_{N_G}\right] \tag{50}
$$

When calculating $\mathbf{R}_{\mathbf{yy}}$ (49), $\mathbf{H}^{(j)}$ is also used in place of **H**. This can also be thought of as setting the user's variance $\sigma_{\mathrm{b}}^{2}$ to 0 for the detected users, since we assume they were detected correctly. Therefore, $\mathbf{R}_{\mathbf{ee}}$ becomes smaller, and only reflects the error

variance of the remaining users as the algorithm proceeds through the groups. The EVM technique for the $j^{th}$ group is then

$$\mathbf{R}_{ee} = \sigma_b^2 \left( \mathbf{I}_{N-(j-1)G} - \sigma_b^2 \mathbf{H}^{(j)\dagger} \mathbf{R}_{yy}^{(j)^{-1}} \mathbf{H}^{(j)} \right)$$
$$\text{where } \mathbf{R}_{yy}^{(j)} = \sigma_b^2 \mathbf{H}^{(j)} \mathbf{H}^{(j)\dagger} + N_0 \mathbf{I}_M \tag{51}$$

Between iterations, **H** is reset to include all users. It should be remembered that as the users are sorted into groups, the columns of **H** must be permuted to match. Also, the group ordering must be completed between each group detection.

## 3.4 Interference Cancellation between Groups

Between groups, IMUD uses soft IC to remove the information that we have determined from the last group's results. The soft IC is accomplished with the use of the mean and variance of each user symbol, which are derived from the APPs in the following paragraphs. Unlike the hard IC techniques used in [8] and [9], soft IC does not make hard cancellation errors, in which subtraction of erroneous symbol decisions actually doubles the interference contribution instead of removing it. If the detector is uncertain and returns an APP of 0 (a probability of ½ for either BPSK symbol), then the mean of that symbol is 0 and no IC takes place.

To further improve the performance of the system, the symbol variance is used for the detection procedure, and is used to weight the channel states appropriately. If the detection for a user was completely uncertain, then our resulting mean will be zero and there will effectively be no IC for that user. Our use of the detected user APPs (denoted as $\Pr(\mathbf{b} = \mathbf{k})$ for all $N$ users or $\Pr(b_{ij} = k)$ for the $i^{th}$ user of the $j^{th}$ group) is derived next. As mentioned previously, IMUD is derived for a BPSK system. However, conversion of the BPSK IMUD to an MPSK system would be straightforward.

The mean for the $l^{th}$ user of the $j^{th}$ group is

$$\begin{aligned}\mu_{ij} &= \mathrm{E}\left[b_{ij}\right] = \sum_{k=-1,+1} k \cdot \Pr\left[b_{ij} = k\right] \\ &= (+1)\Pr\left[b_{ij} = +1\right] + (-1)\Pr\left[b_{ij} = -1\right] \\ &= \Pr\left[b_{ij} = +1\right] - \left(1 - \Pr\left[b_{ij} = +1\right]\right) \\ &= 2\Pr\left[b_{ij} = +1\right] - 1\end{aligned} \tag{52}$$

Using (52), the variance of the user is

$$\begin{aligned}\sigma_{ij}^2 &= \mathrm{E}\left[\left(b_{ij} - \mathrm{E}\left[b_{ij}\right]\right)^2\right] = \mathrm{E}\left[b_{ij}^2\right] - \mathrm{E}^2\left[b_{ij}\right] \\ &= \sum_{k=-1,+1} k^2 \cdot \Pr\left[b_{ij} = k\right] - \mathrm{E}^2\left[b_{ij}\right] \\ &= \Pr\left[b_{ij} = +1\right] + \Pr\left[b_{ij} = -1\right] - \mathrm{E}^2\left[b_{ij}\right] \\ &= 1 - \mu_{ij}^2\end{aligned} \tag{53}$$

For the soft IC, we use the mean of the APP from the previous group in place of a decision. After detecting the $j^{th}$ group, the sample vector for the next group becomes

$$\begin{aligned}\mathbf{y}^{(j+1,1)} &= \mathbf{y} - \sum_{i=1}^{j} \mathbf{H}_i \boldsymbol{\mu}_i \\ &= \mathbf{y}^{(j,1)} - \mathbf{H}_j \boldsymbol{\mu}_j\end{aligned} \tag{54}$$

where $\boldsymbol{\mu}_j$ is

$$\boldsymbol{\mu}_j = \begin{pmatrix} \mu_{1j} \\ \mu_{2j} \\ \vdots \\ \mu_{Gj} \end{pmatrix} \tag{55}$$

$\mathbf{y}^{(j+1,1)}$ is then used in the GAPP detection for the next group.

The variance of the $l^{th}$ user of the $j^{th}$ group is used in GAPP as follows. $\sigma_{ij}^2$ for the current group is put into a diagonal matrix, $\boldsymbol{\Sigma}_j$

$$\boldsymbol{\Sigma}_j = \begin{pmatrix} \sigma_{1j}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{2j}^2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{Gj}^2 \end{pmatrix} \tag{56}$$

Note that for users that have not yet been detected, $\boldsymbol{\Sigma}_j = \mathbf{I}_G$. This variance matrix for the first iteration (which does not include the covariances since we calculated the statistics of the users individually) modifies (39) as follows

$$\begin{aligned} \mathbf{R}_U^{(j,1)} &= \mathrm{E}\left[\left(\mathbf{y}^{(j,1)} - \mathbf{H}_j\mathbf{b}_j\right)\left(\mathbf{y}^{(j,1)} - \mathbf{H}_j\mathbf{b}_j\right)^\dagger \,|\, \mathbf{H}, \mathbf{b}_j\right] \\ &= \mathrm{E}\left[\left(\mathbf{y} - \sum_{i=1}^{j-1}\mathbf{H}_i\boldsymbol{\mu}_i - \mathbf{H}_j\mathbf{b}_j\right)\left(\mathbf{y} - \sum_{i=1}^{j-1}\mathbf{H}_i\boldsymbol{\mu}_i - \mathbf{H}_j\mathbf{b}_j\right)^\dagger \,|\, \mathbf{H}, \mathbf{b}_j\right] \\ &= \sum_{i=1}^{j-1}\mathbf{H}_i\mathrm{E}\left[(\mathbf{b}_i - \boldsymbol{\mu}_i)(\mathbf{b}_i - \boldsymbol{\mu}_i)^\dagger\right]\mathbf{H}_i^\dagger + \sum_{i=j+1}^{N_G}\mathbf{H}_i\mathrm{E}\left[\mathbf{b}_i\mathbf{b}_i^\dagger\right]\mathbf{H}_i^\dagger + \mathrm{E}\left[\mathbf{n}\mathbf{n}^\dagger\right] \\ &= \sum_{\substack{i=1 \\ i\neq j}}^{N_G}\mathbf{H}_i\boldsymbol{\Sigma}_j\mathbf{H}_i^\dagger + N_0\mathbf{I} \end{aligned} \tag{57}$$

since $\mathbf{y}^{(j,\gamma)}$ is zero-mean (the means of all the detected users have been removed) and $\boldsymbol{\Sigma}_i$ for undetected groups is the identity matrix. For subsequent iterations, all symbols have means and variances from the previous iteration and therefore the second term of the third line in (57) is non-existent.

To incorporate the previous iteration's means and variances into the notation for group mean and variance, the vector notation is altered slightly. Denote the previous iterations means as

$$\boldsymbol{\mu}_j^{(\gamma)} = \begin{pmatrix} \mu_{1j} \\ \mu_{2j} \\ \vdots \\ \mu_{Gj} \end{pmatrix} \tag{58}$$

where $\gamma$ references the iteration and $\mu_{ij}$ is the $i^{th}$ user of the $j^{th}$ group of the new random ordering. Keep in mind that in each iteration, the user denoted by *ij* is different since users are reordered between iterations. Similarly, for the variances,

$$\mathbf{\Sigma}_j^{(\gamma)} = \begin{pmatrix} \sigma_{1j}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{2j}^2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{Gj}^2 \end{pmatrix} \tag{59}$$

For the $\gamma^{th}$ iteration, the matrices $\boldsymbol{\mu}^{(\gamma-1)}$ and $\mathbf{\Sigma}^{(\gamma-1)}$ are used initially as the means and variances for all the users in equations (54) and (57) with one exception. The current group under detection does not have its mean removed from the measurement vector $\mathbf{y}$, nor are the users symbol variances used in the GAPP detection. It is detrimental to the receiver to cancel the group under detection; we simply want to remove the interference as much as possible to allow for a good decision. After the detection is complete, the new means and variances of that group replace the previous means and variances in $\boldsymbol{\mu}^{(\gamma)}$ and $\mathbf{\Sigma}^{(\gamma)}$. For the next group, the updated values are used in the mean subtraction and group MAP detection. This effectively changes (54) to

$$\mathbf{y}^{(j+1,\gamma)} = \mathbf{y} - \sum_{\substack{i=1 \\ i \neq j+1}}^{N_G} \mathbf{H}_i \boldsymbol{\mu}_i^{(\gamma)} \tag{60}$$

and (57) to

$$\mathbf{R}_U^{(j,\gamma)} = \sum_{\substack{i=1 \\ i \neq j}}^{N_G} \mathbf{H}_i \mathbf{\Sigma}_j^{(\gamma)} \mathbf{H}_i^t + N_0 \mathbf{I} \tag{61}$$

As mentioned above, the covariances between detected symbols are not determined since the symbols are treated as independent in every iteration. The EVM technique described above uses MMSE suppression as a way of determining error

variance. It is ineffective in finding the covariances since each user symbol variance is found by suppressing all other users. If ML were to be used, the covariances could be found; however, that would undermine the point behind IMUD, which is a technique with fewer computations than the optimum JML.

## 3.5 Iterations

The performance improvement of IMUD between iterations is due primarily to the interleaving of the users. For a single iteration of IMUD, gains over the group detection of Ng and Sousa [9] are a result of the user ordering and soft interference cancellation. However, if the same ordering were to be used for every iteration, there would be no benefit. This is because there would be no source of added information for the detector; the groups are identical, with the *a priori* information duplicating the APPs out of the detector. Therefore, interleaving of the users between iterations is necessary. It is a means for any disadvantaged users from the previous iteration to have a second opportunity to be included in more favourable groups. The users will experience different correlations.

As introduced in chapter 2, LLRs are used throughout much of the literature in Turbo and serially concatenated codes. However, because IMUD uses the probabilities of the symbols to compute the mean and variance, and the symbol probabilities are necessary for the marginalization in GAPP, it is simpler in IMUD to use symbol probabilities. It is straightforward to convert LLRs to symbol probabilities (36) to allow easy interconnection between decoder.

## 3.6 The IMUD Process

The IMUD process is summarized here:

1. Set $\boldsymbol{\mu}$ to an zero length $N$ vector and $\boldsymbol{\Sigma}$ to an $N \times N$ identity matrix

2. Initialize $j$ and $\gamma$ to 1

3. **For** $\gamma = 1{:}iter$,

   3.1 **For** $j = 1{:}N_G$,

      3.1.1 If $\gamma = 1$, sort all undetected users with EVM sort using (51) and form group $j$

      3.1.2 Find group $j$ users joint APP values with (43) using (60) and (61)

      3.1.3 Calculate the APPs ($\Pr\left(b_{ij} \mid \mathbf{y}\right)$) from group $j$ with (45)

      3.1.4 Calculate the means and variance for group $j$ with (52) and (53), respectively

      3.1.5 Update $\boldsymbol{\mu}_j^{(\gamma)}$ and $\boldsymbol{\Sigma}_j^{(\gamma)}$

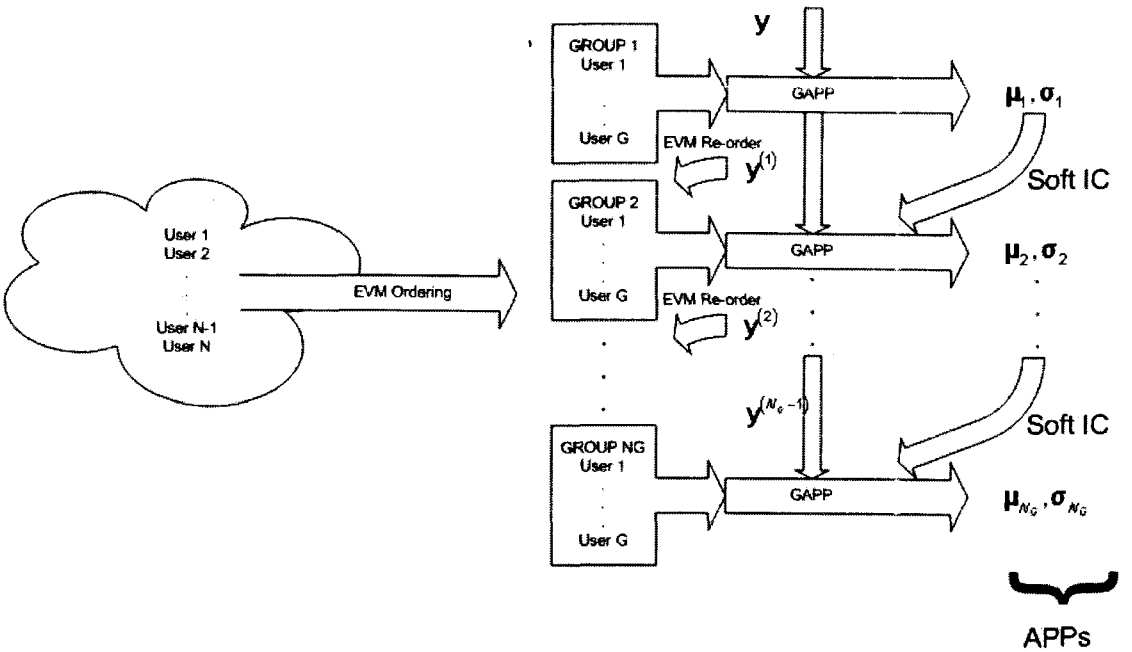
      3.1.6 Calculate $\mathbf{y}^{(j,\gamma)}$ with (60)

   3.2 Randomize the user order

4. Make hard decision for each user using the APPs

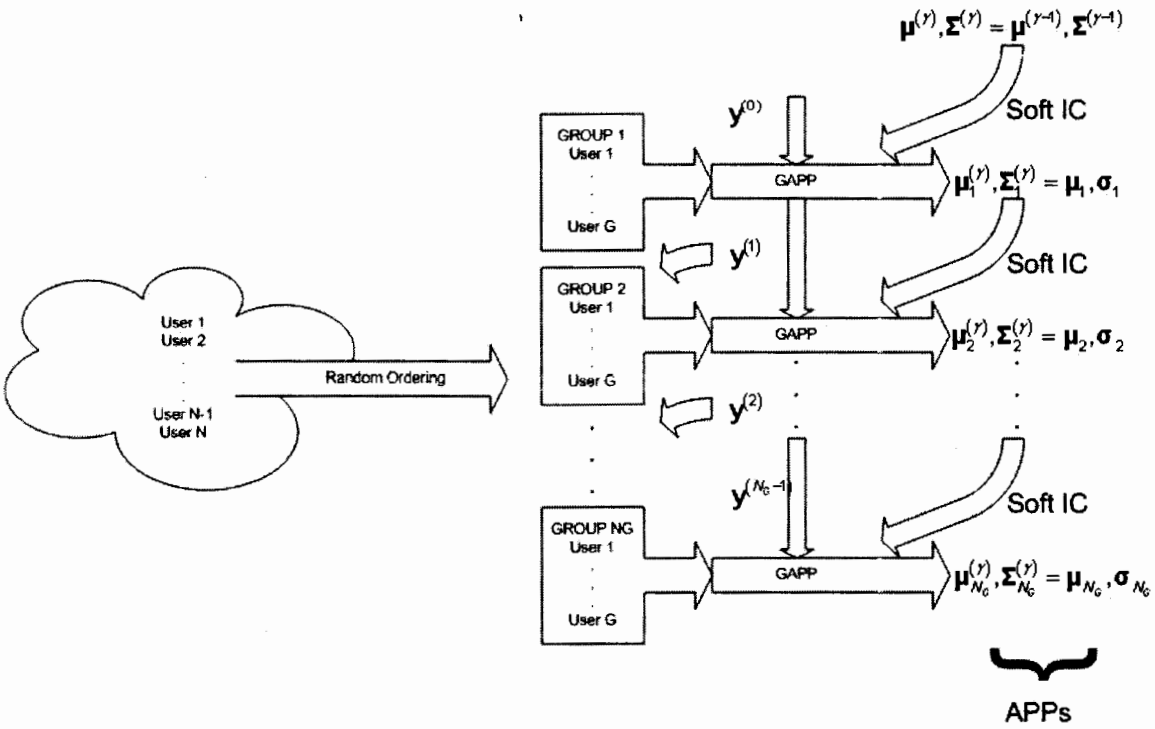
The first stage of IMUD detection begins as follows. All users are initially sorted with (51). The top $G$ users (the $G$ with the lowest error variance) make up the 1st group. The first group's likelihoods are calculated with (43), where $\mathbf{y}^{(1,1)}$ is the same as the initial measurement vector $\mathbf{y}$. $\mathbf{R}_U^{(1,1)}$ is found with (61). Using the APPs generated with (45), the means and variances from (52) and (53) are calculated. $\mathbf{y}^{(2,1)}$ is formed with (60), using the new means $\boldsymbol{\mu}_1^{(1)}$ for group 1. The means for all other users are still zero. The detected groups channels are removed from the $\mathbf{H}$ matrix used for ordering, and the remaining users are sorted with (51). The process continues with the detection of the 2nd group and the formation of $\mathbf{y}^{(3,1)}$, and continues until the detection of all $N_G$ groups is complete. The process for the first iteration is shown in Figure 11.

**Figure 11: First iteration of IMUD**




For the second iteration of IMUD (Figure 12), there are a few key changes from the first iteration. First, the EVM ordering is replaced with a random ordering. As long as the user order is different (the user group are significantly different) than the original EVM sort, there is no observable difference in performance, as shown in Section 3.8.4. The soft IC is still done and the previous iteration's means $\boldsymbol{\mu}^{(\gamma-1)}$ and variances $\boldsymbol{\Sigma}^{(\gamma-1)}$ are used as a starting point. At the beginning of the $J^{th}$ iteration, the means and variances are preset so that $\boldsymbol{\mu}^{(\gamma)} = \boldsymbol{\mu}^{(\gamma-1)}$ and $\boldsymbol{\Sigma}^{(\gamma)} = \boldsymbol{\Sigma}^{(\gamma-1)}$.

**Figure 12: Subsequent iterations of IMUD**




## 3.7 Complexity

The complexity of the MUD techniques covered so far can be split into two categories: complexity of the detection and of the ordering. The complexity for JML, MMSE, GD, group ML, MMSE V-BLAST and IMUD will be compared.

As mentioned in Section 2, the channel gains must be updated frequently in a fast fading channel. This translates to the need to recalculate filters as the channel changes. Since the IMUD simulations use a fast fading channel, it is assumed that the channel gains are updated every symbol. However, in a quasi-static channel, the channel gains, and thus the filters, are valid for many symbols. The following analysis focuses on the fast fading channel; Table 2 in section 3.7.8 shows the difference in complexity for the techniques between the fast and quasi-static channel.

First, a few commonalities for the analysis of complexity are reviewed. The unit used to represent complexity is that of a single complex multiply and add procedure (CMA), common to microprocessors. For example, when taking the product of **Hb** where **H** is $M \times N$ and **b** is $N \times 1$, the processor would need to compute *MN* CMAs. Also, the complexities derived for each of the techniques can contain many terms due to all the matrix operations. When reporting the complexity of a technique, the lower order terms are neglected.

A common procedure used in the MUD techniques is that of a matrix inversion. Using standard row reduction to compute the matrix inverse, the complexity of inverting an $M \times M$ matrix is approximately equal to $M^3$. For further computational savings, updates to the inverse can be accomplished using Woodbury's Identity [7]. The matrix update can be accomplished between groups or between users, when the contribution of the a group or user needs to be removed from the inverse. Users and groups can also be added using the Woodbury's Identity; this is ideal for the undesired covariance matrix $\mathbf{R}_U$, where groups are both added and removed between APP extractions. The complexity is as follows: for all groups in a groupwise system, the covariance inverse update is $M^3 + GM(N_G - 1)(2M + 1)$, where the first term is for the initial inverse and the following term is for all subsequent group updates. The computational savings for a 12 user and 12 antenna system using groups containing 2 users is 4728 CMAs compared to 10368 CMAs without matrix updates. For single user removal or addition, the complexity is $M^3 + M(N-1)(2M+1)$. With modifications to allow for the specifics of each technique, these complexity terms are used throughout section 3.7.

Since most of the detection and ordering techniques depend on the inverse of the covariance of the measurement vector, the complexity is derived here. $\mathbf{R}_{yy}$ is from

(24) for MMSE V-BLAST and GD, and requires $M^2N$ CMAs. $\mathbf{R}_U$ is from (39) for IMUD, and requires $M^2(N-G)$ CMAs. Both covariance matrices are $M \times M$, so they require approximately $M^3$ CMAs for an inversion.

Finally, for the techniques that use interference cancellation, the channel gain matrices shrink as users are removed from the system. This affects the ordering techniques. Instead of $\mathbf{H}$ being a static $M \times N$ matrix, it becomes dependent on the group or user under detection, as shown in $\mathbf{H}^{(j)}$ (50). Therefore, when computing a matrix multiplication such as $\mathbf{H}^{(j)}\mathbf{H}^{(j)\dagger}$, the complexity for all $N_G$ groups is not $M^2NN_G$ but $M^2\left[\sum_{k=0}^{N_G-1}(N-kG)\right]$. For the single user ordering techniques used for V-BLAST, such a computation would change from $M^2N^2$ to $M^2\left[\sum_{k=0}^{N-1}(N-k)\right]$. These summations will be referred to as

$$S_1 = \sum_{k=0}^{N_G-2}(N-kG) \tag{62}$$

for group ordering and

$$S_2 = \sum_{k=0}^{N-2}(N-k) \tag{63}$$

for single-user ordering. The final group or user inverse does not need to be calculated since the outcome is obvious (there are only $G$ users left for the last group). The last summation, $S_3$, is used to calculate the computational load of evaluating $\mathbf{R}_U$ for the Group Detection technique.

$$S_3 = \sum_{k=1}^{N_G}(N-kG) \tag{64}$$

Notice that for the last group, the complexity for the inverse is 0 (the summation is 0). This is because there are no interferers, and thus the covariance matrix is simply the identity matrix.

Summations $S_1$ through $S_3$ have simple closed forms. For this discussion however, it is clearer to leave them as explicit summations.

### 3.7.1 Joint Maximum Likelihood MUD Complexity

For JML, there is no ordering complexity since all user decisions are made simultaneously. Hence the complexity is only due to the joint detection. Using the expression for JML from (9),

$$\underbrace{\left.\begin{array}{r}\mathbf{Hb} \rightarrow MN \text{ CMAs} \\ \left(\mathbf{y}-(\mathbf{Hb})\right)^{\dagger}\left(\mathbf{y}-(\mathbf{Hb})\right) \rightarrow M \text{ CMAs}\end{array}\right\} \text{for all comb. of } \mathbf{b}}_{\text{for a total complexity of } 2^{N}(M+MN)} \tag{65}$$

The complexity can be reduced to $2^{N+1}M$ if Gray coding is used when enumerating over $\mathbf{b}$. This comes from being able to update $\mathbf{y}-\mathbf{Hb}$ for the bit that is different between each successive $\mathbf{b}$ by simply adding the appropriate channel. The reduced complexity of Gray coding will be used for JML and the group techniques. Considering that the covariance matrix is not diagonal for the group techniques, the complexity of joint detection becomes

$$2^{G+1}\left(M^2+M\right) \tag{66}$$

### 3.7.2 MMSE MUD Complexity

The linear techniques are the most computationally least expensive out of all the MUDs. The highest complexity in MMSE MUD lies in the formation of the filter. For MMSE MUD, the complexity using (21) and (23) is

$$\underbrace{\left.\begin{array}{l}\mathbf{R}_{yy}^{-1} \rightarrow M^2N + M^3 \text{ CMAs} \\ \mathbf{R}_{yy}^{-1}\mathbf{H} \rightarrow M^2N \text{ CMAs}\end{array}\right\}\text{filter calculation}}_{\left.\mathbf{W}^{\dagger}\mathbf{y} \rightarrow MN \text{ CMAs}\right\}\text{includes all users}} \qquad (67)$$

$$\text{total complexity of } M\left(M^2+2MN+N\right)$$

### 3.7.3 MMSE Group Detection MUD Complexity

For the MMSE GD of Ng and Sousa [9], the complexity for the GD and filtering will now be examined in the following. The interest in discovering the complexity of MMSE GD is in its comparison to GML. Therefore, the use of the reliability metric (detailed in [9]) is not examined, only the detection itself. Using the expressions from (30), (31) and (32) and including hard IC between groups, the complexity of MMSE GD is

$$\underbrace{\left.\begin{array}{l}\mathbf{R}_{yy}^{-1} \rightarrow M^2N + M^3 + GM\left(N_G - 1\right)\left(2M+1\right) \text{ CMAs} \\ \mathbf{Q}_j^{-1} = \left(\mathbf{W}_j^{\dagger}\mathbf{R}_U\mathbf{W}_j\right)^{-1} \rightarrow M^2S_3 + \left(N_G - 1\right)\left(GM^2 + G^2M + G^3\right) \text{ CMAs}\end{array}\right\}\text{for all groups}}_{\left.\begin{array}{r}\mathbf{W}_j = \mathbf{R}_{yy}^{-1}\mathbf{H}_j \rightarrow GM^2 \text{ CMAs} \\ \mathbf{W}_j^{\dagger}\mathbf{H}_j \rightarrow G^2\text{M CMAs} \\ \mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} - \mathbf{H}_j\hat{\mathbf{b}}_j \rightarrow GM \text{ CMAs} \\ \mathbf{x}_j = \mathbf{W}_j^{\dagger}\mathbf{y}^{(j)} \rightarrow GM \text{ CMAs} \\ \left(\mathbf{x}_j - \mathbf{W}_j^{\dagger}\mathbf{H}_j\mathbf{b}_j\right)^{\dagger}\mathbf{Q}_j^{-1}\left(\mathbf{x}_j - \mathbf{W}_j^{\dagger}\mathbf{H}_j\mathbf{b}_j\right) \text{ (65)} \rightarrow 2^{G+1}\left(G^2+G\right) \text{ CMAs}\end{array}\right\}\text{once per group}} \qquad (68)$$

$$\text{total complexity of } \sim \left(M^2N+M^3+2GM^2\left(N_G-1\right)\right)+\left(M^2S_3+\left(N_G-1\right)\left(GM^2+G^2M+G^3\right)\right)+GN_G\left(M^2+GM+2M+2^{G+1}G\right)$$

Similar to JML, Gray coding is used to remove some of the complexity associated with the calculation of $\left(\mathbf{W}_j\mathbf{H}_j\right)\mathbf{b}_j$ for every $\mathbf{b}_j$. Since hard IC is used after a group is detected, the users of that group does not appear in subsequent groups. This means that the inverse of $\mathbf{R}_{yy}$ must be updated for every group. For $\mathbf{Q}_j$, the update is more complex since $\mathbf{R}_U$ is multiplied by $\mathbf{W}$, but as groups are detected, the complexity of the inverse does get smaller. If IC were to be removed, $\mathbf{R}_{yy}^{-1}$ would only need to be

calculated once per symbol since the user channel gains would no longer be removed from the **H** matrix during recalculation; however, $\mathbf{Q}_j^{-1}$ would not reduce in complexity as the groups are detected.

### 3.7.4 MMSE Group ML MUD Complexity

The BER performance of MMSE GML is identical to MMSE GD, as discussed previously. The complexity is similar to MMSE GD, except for the complexity of the matrix inverses. From (38), (39), and (41), the complexity is found as

$$\underbrace{\begin{array}{l}\mathbf{R}_U^{-1} \rightarrow M^2(N-G)+M^3+GM(N_G-1)(2M+1)\ \text{CMAs}\Big\}\text{includes all groups}\\ \left.\begin{array}{l}\mathbf{y}^{(j+1)}=\mathbf{y}^{(j)}-\mathbf{H}_j\hat{\mathbf{b}}_j \rightarrow GM\ \text{CMAs}\\ \left(\mathbf{y}-\mathbf{H}_j\mathbf{b}_j\right)^{\dagger}\mathbf{R}_U^{-1}\left(\mathbf{y}-\mathbf{H}_j\mathbf{b}_j\right)\ (65) \rightarrow 2^{G+1}\left(M^2+M\right)\ \text{CMAs}\end{array}\right\}\text{once per group}\end{array}}_{\text{total complexity of} \sim \left(M^2(N-G)+M^3+2GM^2(N_G-1)\right)+MN_G\left(G+2^{G+1}M\right)} \quad (69)$$

$\mathbf{H}_j\mathbf{b}_j$ is not included in the complexity since Gray coding is used. The GML computes only the inverse of the undesired variance matrix $\mathbf{R}_U$. $\mathbf{R}_{yy}$ is not calculated at all, and this allows for a significant computational savings over GD. Intuitively, this savings can be seen in the fact that even though GD and GML have the same results, GD must calculate a filter for every group, increasing the computational load. This is why the soft-output version of GML is used in IMUD. The computational difference between GML and GAPP is negligible in comparison to the matrix computations necessary in this BPSK system. GAPP only requires extra additions of the log probabilities. With larger constellations, the complexity of GAPP may be significantly increased to complete the marginalization.

### 3.7.5 Ordering Complexity

The ordering complexity will be calculated separately for V-BLAST and IMUD. For random ordering, the complexity is thought of as free in these simulations, but in a real system, some sort of randomizing algorithm must be used and included in complexity calculations. The three non-random ordering schemes compared include the SNR sort, LLR sort and the EVM sort. Each is derived in Section 3.8.4 during the performance discussion.

#### 3.7.5.1 SNR Ordering

For ordering by SNR, the ZF filter is used to suppress all other users. The ZF filter is calculated just as in (19). From (75)

$$\underbrace{\left.\begin{array}{l}\mathbf{H}\left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1} \rightarrow 2MS_1^2 + S_1^3 \text{ CMAs} \\ \hline \mathbf{H}\left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1} \rightarrow 2MS_2^2 + S_2^3 \text{ CMAs} \\ \mathbf{W}\mathbf{W}^{\dagger} \rightarrow S_1M^2 \text{ CMAs} \\ \hline \mathbf{W}\mathbf{W}^{\dagger} \rightarrow S_2M^2 \text{ CMAs}\end{array}\right\} \begin{array}{l}\text{groupwise, for all groups} \\ \hline \text{single-user, for all users}\end{array}}_{\text{total complexity of} \sim \frac{S_1^3 + S_1^2 2M + S_1M^2 \text{ for groups}}{S_2^3 + S_2^2 2M + S_2M^2 \text{ for users}}} \tag{70}$$

It should be noted that for some of these techniques, the complexity can be shared among the components. For example, if using ZF V-BLAST with SNR sorting, the filter for each user's detection is already calculated during the sorting algorithm.

#### 3.7.5.2 LLR Ordering

For log-likelihood ratio (LLR) ordering, (78), the detection depends on a ZF filter implicitly. The only difference is that the SNR calculated by the ZF filter is scaled by $\left|\mathrm{Re}\left(\mathbf{W}^{\dagger}\mathbf{y}\right)\right|$. The complexity is

$$\underbrace{\left.\begin{array}{l}\text{complexity for groups/users from (69)}\\ \mathbf{W}^{\dagger}\mathbf{y} \rightarrow S_1 M \text{ CMAs}\\ \hline \mathbf{W}^{\dagger}\mathbf{y} \rightarrow S_2 M \text{ CMAs}\end{array}\right\} \frac{\text{groupwise, for all groups}}{\text{single-user, for all users}}}_{\text{total complexity of} \sim \frac{S_1^3+S_1^2 2M+S_1(M^2+M) \text{ for groups}}{S_2^3+S_2^2 2M+S_2(M^2+M) \text{ for users}}} \tag{71}$$

### 3.7.5.3 EVM Ordering

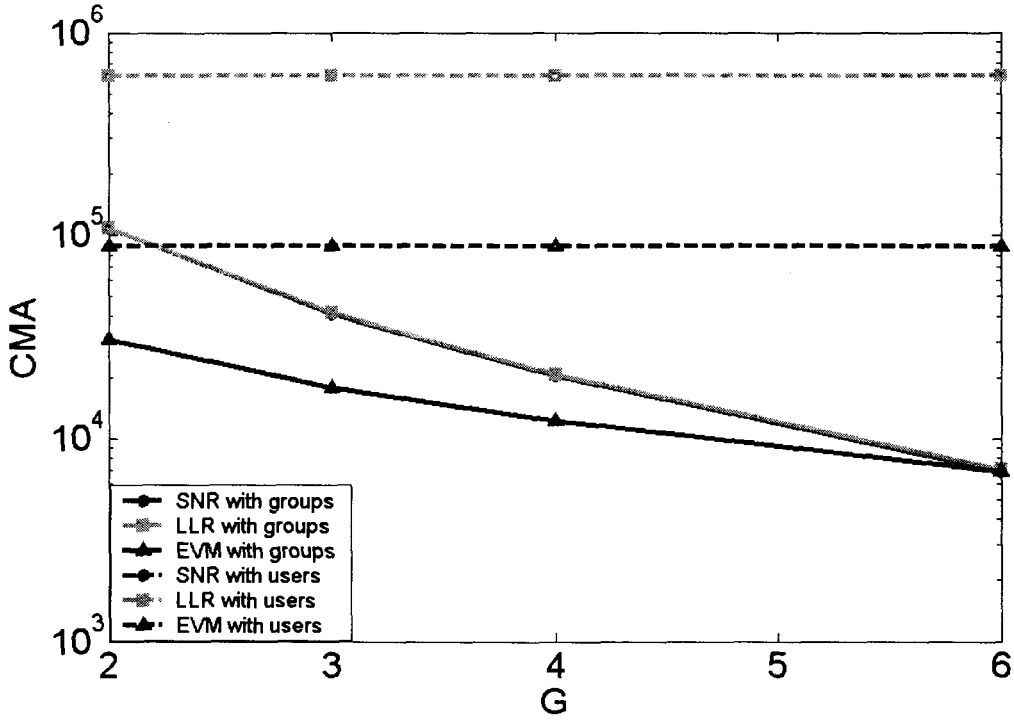
Finally, EVM ordering; notice that $\mathbf{R}_{yy}$ is updated with the Woodbury Identity. From (51), the complexity is

$$\underbrace{\left.\begin{array}{l}\mathbf{R}_{yy}^{-1} \rightarrow M^2N + M^3 + GM(N_G - 2)(2M+1) \text{ CMAs}\\ \hline \mathbf{R}_{yy}^{-1} \rightarrow M^2N + M^3 + M(N-2)(2M+1) \text{ CMAs}\\ \left(\mathbf{R}_{yy}^{-1}\right)\mathbf{H}^{(j)} \rightarrow M^2 S_1 \text{ CMAs}\\ \hline \left(\mathbf{R}_{yy}^{-1}\right)\mathbf{H}^{(j)} \rightarrow M^2 S_2 \text{ CMAs}\\ \mathbf{H}^{(j)\dagger}\left(\mathbf{R}_{yy}^{-1}\mathbf{H}^{(j)}\right) \rightarrow MS_1^2 \text{ CMAs}\\ \hline \mathbf{H}^{(j)\dagger}\left(\mathbf{R}_{yy}^{-1}\mathbf{H}^{(j)}\right) \rightarrow MS_2^2 \text{ CMAs}\end{array}\right\} \frac{\text{includes all groups}}{\text{includes all users}}}_{\text{total complexity of} \sim \frac{M^2N+M^3+2GM^2(N_G-2)+MS_1^2+M^2S_1 \text{ for groups}}{M^2N+M^3+2M^2(N-2)+MS_2^2+M^2S_2 \text{ for users}}} \tag{72}$$

### 3.7.5.4 Ordering Technique Comparison

The ordering techniques are similar for both sorting in groups and users. The exception is that when computing the inverse, it must be done nearly $N - N_G$ more times for the single user case ($N-1$ inverses calculated for single user ordering, $N_G - 1$ inverses calculated for group ordering). A comparison of the sorting techniques for groups and for users in shown in Figure 13.

**Figure 13: Comparison of the approximate complexity for ZF, LLR and EVM ordering techniques for 12 user/12 receive antenna system**




The figure shows the interesting fact that all the ordering techniques have roughly the same complexity. In fact, EVM is slightly less complex than SNR and LLR ordering since it uses the linear filter implicitly and does not need to actually calculate the filter. EVM ordering also uses the Woodbury Identity to update the covariance matrix. As expected, the total complexity for group ordering is less than that of user ordering. Also, as the group size increases, the group order complexity reduces. This is due to the matrix inverse complexity; for more groups of smaller sizes, the number of total matrix inverses is large. The complexity of doing more inverses is more than that of doing larger inverses in this case.

### 3.7.6 MMSE V-BLAST MUD Complexity

MMSE V-BLAST was chosen as a technique to compare against IMUD because of its popularity and performance. For MMSE V-BLAST with EVM sorting, the

complexity is essentially that of MMSE MUD plus the EVM ordering technique and the recalculation of $\mathbf{R}_{yy}$ for each user. The complexities in the fast fading and quasi-static cases are quite different; for MMSE V-BLAST in a static channel, the filter and the ordering need only be calculated once. The complexity using (21), (23), (25), and (72) is

$$\underbrace{\begin{array}{l}\left.\begin{array}{l}\mathbf{R}_{yy}^{-1} \rightarrow M^2N + M^3 + M(N-2)(2M+1) \text{ CMAs} \\ \mathbf{R}_{yy}^{-1}\mathbf{H} \rightarrow M^2N^2 \text{ CMAs}\end{array}\right\} \text{filter creation for all users} \\ \left.\begin{array}{r}MS_2^2 + M^2S_2 \text{ CMAs}\end{array}\right\} \begin{array}{l}\text{EVM ordering for} \\ \text{all users (without } \mathbf{R}_{yy}^{-1}\text{)}\end{array} \\ \left.\begin{array}{r}\mathbf{W}_j^{\dagger}\mathbf{y}^{(j)} \rightarrow M \text{ CMAs} \\ \mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} - \mathbf{h}_j\hat{b}_j \rightarrow M \text{ CMAs}\end{array}\right\} \text{for all users}\end{array}}_{\text{total complexity of} \sim \left(M^2N+M^3+2M^2(N-2)+M^2N^2\right)+\left(MS_2^2+M^2S_2\right)} \tag{73}$$

### 3.7.7 IMUD Complexity

IMUD combines GAPP (45) and the EVM sort (72). Using (72), (44), (60), and (61), the complexity for IMUD in a fast fading channel is

$$\underbrace{\begin{array}{l}\left.\begin{array}{r}\sim M^2N + M^3 + 2GM^2\left(N_G - 2\right) + MS_1^2 + M^2S_1\end{array}\right\} \begin{array}{l}\text{EVM ordering} \\ \text{includes all groups}\end{array} \\ \left.\begin{array}{r}\mathbf{R}_U^{-1} \rightarrow \left(N_G - 1\right)\left(GM^2 + 2GM\left(2M+1\right)\right) + M^3 \text{ CMAs}\end{array}\right\} \text{includes all group} \\ \left.\begin{array}{r}\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} - \mathbf{H}_j\hat{\mathbf{b}}_j \rightarrow MG \text{ CMAs} \\ \left(\mathbf{y}-\mathbf{H}_j\mathbf{b}_j\right)^{\dagger}\mathbf{R}_U^{-1}\left(\mathbf{y}-\mathbf{H}_j\mathbf{b}_j\right)\ (65) \rightarrow 2^{G+1}\left(M^2+M\right) \text{ CMAs}\end{array}\right\} \text{once per group}\end{array}}_{\text{total complexity of} \sim iter\left[\left(M^2N+M^3+2GM^2(N_G-2)+MS_1^2+2M^2S_1\right)+\left(5GM^2(N_G-1)+M^3\right)+MN_G\left(G+2^{G+1}M\right)\right]} \tag{74}$$

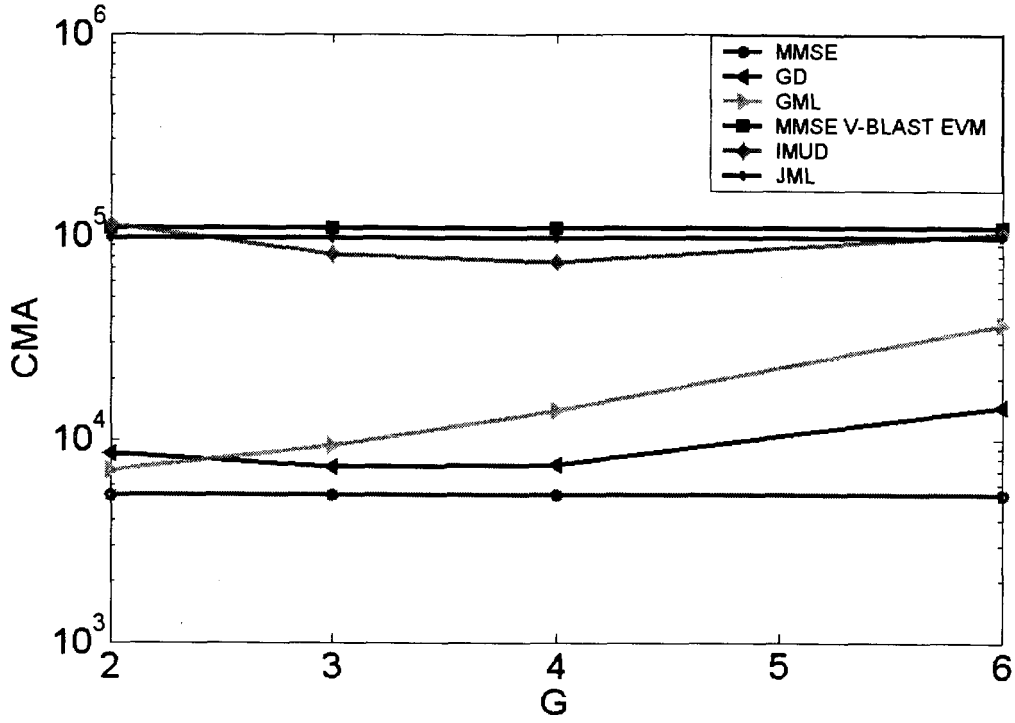
For $\mathbf{R}_U^{-1}$, there is a noticeable increase in complexity over that over GML; there are twice as many matrix inverse updates as before. This is due to the use of the symbol variances (Section 3.4) in GAPP. The actual complexity will most likely be less than that shown in (74) when variances are very small, but this is used as an upper bound. For BPSK, there are $\sim G2^G$ adds and table-lookups (from marginalization of the probabilities and the Jacobian algorithm) that are neglected in the above complexity. For higher

density constellations, these MAP computations become relevant. Also, note that the entire complexity must be taken over the number of iterations run, referred to as *iter*.

### 3.7.8 Comparison of MUD Complexity

Table 2 shows the complexity of each of the preceding techniques. The expressions for complexity in a quasi-static fading channel are the results from the fast fading case with all unnecessary inverse computations removed. Some inverses are unnecessary since the channel is static between symbols, and therefore the MMSE filters and the order of users will stay the same between symbols. 2 iterations of IMUD are run since the performance curves of Figure 26 show that this is enough to allow the technique to converge. Comparing the above equations for fast fading systems with 12 and 24 antennas and various group sizes, we get the following complexity graph in CMAs.

**Figure 14: Comparison of the approximate complexity for GD, GML, MMSE V-BLAST, IMUD(2) and JML for 12 user/12 receiver system in a fast fading channel**

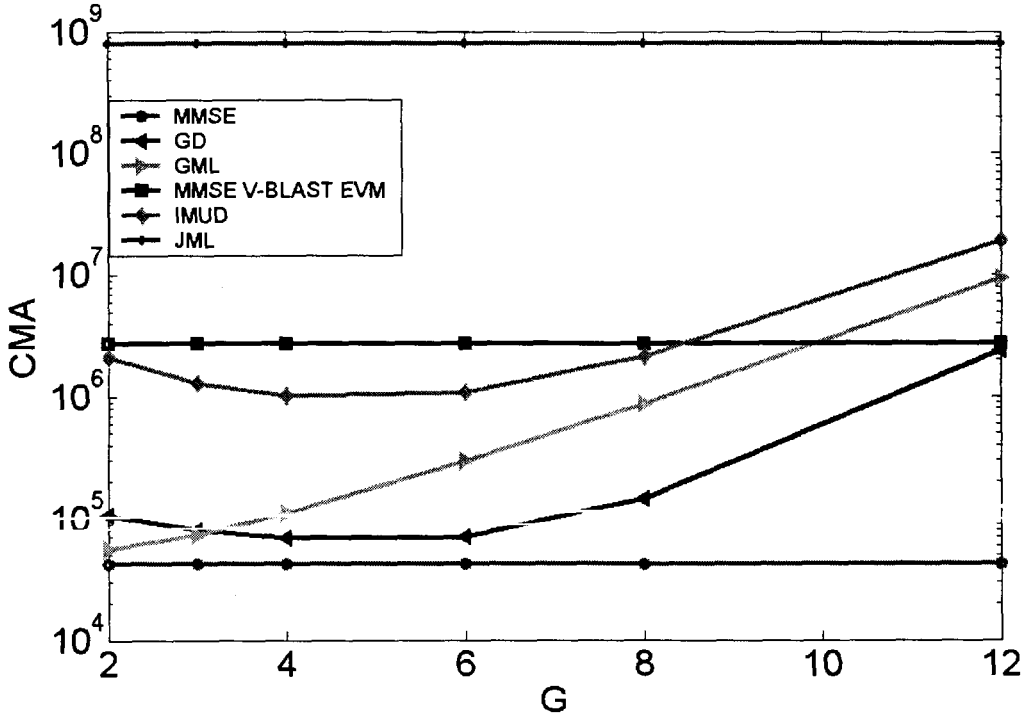



The first point of interest in Figures 14 and 15 is that JML in the 12 user system has complexity comparable to IMUD and MMSE V-BLAST. This is a benefit of the Gray coding; without it, the JML complexity is almost a magnitude higher. However, JML's exponential growth makes its complexity far higher that that of the other techniques in the 24 user system of Figure 15. Another interesting point is that IMUD has more complexity than MMSE V-BLAST for certain system configurations. For example, in the 24 user system the group techniques exhibit a parabolic characteristic such that IMUD performs with more complexity with a group size of 12.

It is somewhat intuitive to think that IMUD will perform with lower complexity with smaller groups due to the smaller size of the GAPP detection. However, due to the complexity of the matrix inversion, this is not always true. In the 12 antenna system, the lowest IMUD complexity is attained with groups of 4 users, which corresponds to a lower

complexity (Figure 13) but higher performance (Figures 18 and 19) than IMUD with groups of 3 users. In the 24 antenna system, the minimum IMUD complexity is achieved with groups of 4 users.

**Figure 15: Comparison of the approximate complexity for GD, GML, MMSE V-BLAST, IMUD(2) and JML for 24 user/24 receiver system in a fast fading channel**




For the overloaded case where there are 12 users but only 8 receive antennas, the complexity difference between IMUD and MMSE V-BLAST is increased. Figure 16 shows the results.

**Figure 16: Comparison of the approximate complexity for GD, GML, MMSE V-BLAST, IMUD(2) and JML for 12 user/8 receiver system in a fast fading channel**

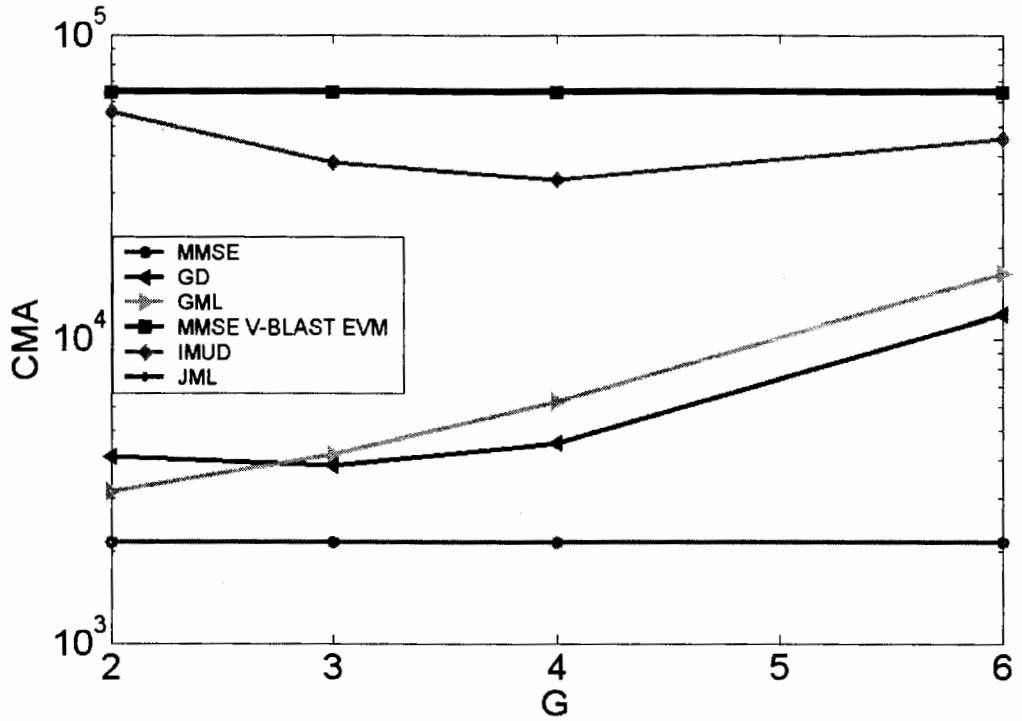


**Table 2: Complexity expressions for MUD techniques**

| Technique | Complex Multiplies/Adds (Fast Fading) | Complex Multiplies/Adds (Quasi-Static Fading) |
|---|---|---|
| **Joint Maximum Likelihood (2.4.1, [3])** | $\sim 2^N M(N+1)$ | $\sim 2^N M(N+1)$ |
| **MMSE MUD (2.4.4, [6])** | $\sim M(M^2+2MN+N)$ | $\sim MN$ |
| **MMSE Group Detection w/hard IC (Section 2.4.6, [4])** | $\sim (M^2N+M^3+\cdots$ $2GM^2(N_G-1))+\cdots$ $(M^2S_3+(N_G-1)(GM^2+\cdots$ $G^2M+G^3))+GN_G(\cdots$ $M^2+GM+2M+2^{G+1}G)$ | $\sim GN_G(2M+2^{G+1}G)$ |
| **Group ML w/hard IC (3.1)** | $(M^2(N-G)+M^3+\cdots$ $2GM^2(N_G-1))+\cdots$ $MN_G(G+2^{G+1}M)$ | $\sim MN_G(G+2^{G+1}M)$ |
| **MMSE V-BLAST (2.4.5, [8])** | $\sim (M^2N+M^3+2M^2(N-2)+\cdots$ $+M^2N^2)+(MS_2^2+M^2S_2)$ | $\sim 2MN$ |
| **IMUD (3.6)** | $\sim iter[(M^2N+M^3+\cdots$ $2GM^2(N_G-2)+MS_1^2+\cdots$ $2M^2S_1)+(5GM^2(N_G-1)+\cdots$ $M^3)+MN_G(G+\cdots$ $+2^{G+1}M)$ | $\sim iter(5GM^2(N_G-1)+\cdots$ $M^3+MN_G(G+2^{G+1}M))$ |

It is interesting to note that a major portion of IMUDs complexity is due to the ordering technique. Possibilities of further complexity reduction may lie in finding a less complex solution to the user order.

## 3.8 Performance

Bit error rate (BER) curves were compiled to aid in the investigation of the performance of IMUD. The techniques discussed in chapter 2 (MMSE in Section 2.4.4, JML in Section 2.4.1, V-BLAST in Section 2.4.5, and GD in Section 2.4.6) were simulated with system parameters as described in chapter 1. An analytical upper bound on BER [4] was used for JML due to the length of simulations. This section begins with an investigation of the performance of IMUD as explained in Section 3.6. Then, the results of simulations used to determine the most effective interference cancellation, ordering, and feedback techniques are reported. The details are covered in Section 3.8.3, 3.8.4, and 3.8.5, respectively.

From this point on, a MIMO system with $N$ users separated into $N_G$ groups of $G$ users transmitting to a receiver with $M$ antennas will be referred to as a $(\boldsymbol{M,N_G,G})$ system. IMUD will be referred to as IMUD(*iter*), where *iter* is the number of iterations completed.

### 3.8.1 IMUD with Equal Numbers of Users and Receivers (M = N)

The performance of IMUD with equal numbers of users and receivers will be examined next.

Figure 17 shows a (**4,2,2**) system with BER curves from MMSE, GD, IMUD and the JML bound. It demonstrates the benefit of IMUD compared with that of GD, and demonstrates the use of MMSE and JML as upper and lower bounds. Since IMUD is

still a groupwise technique, it can do no worse than MMSE (if $G = 1$) and no better than JML (if $G = N$). Notice that after 1 iteration of IMUD, the performance is less than 0.5dB poorer than JML at a BER of $10^{-3}$. The GD depicted in Section 2.4.6 includes the reliability sorting technique and hard IC as reported in [9]. IMUD is denoted as IMUD(*iter*), where *iter* is the number of iterations completed. 2 iterations were found to be sufficient for IMUD to converge in the systems tested (Section 3.8.3).

**Figure 17: (4,2,2) system**

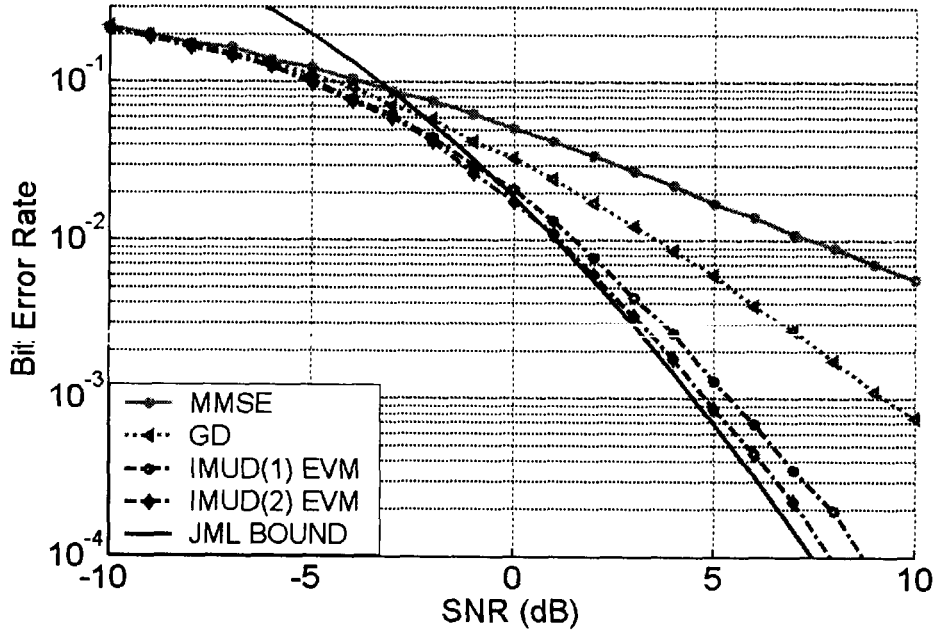



Figure 18 through 21 shows a 12 antenna system with 12 users in four possible group configurations. In a (**12,2,6**) configuration and at a BER of $10^{-3}$, IMUD outperforms MMSE V-BLAST by 2 dB. It also requires a similar computational complexity as MMSE V-BLAST as shown in the complexity graph of Figure 14.

Figure 18: (12,2,6) system

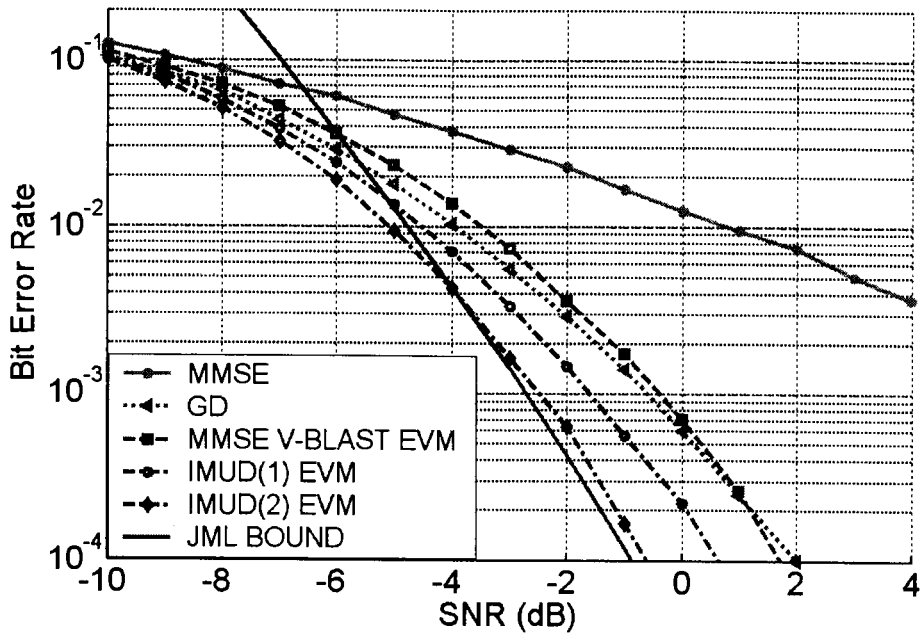



Figure 19: (12,3,4) system

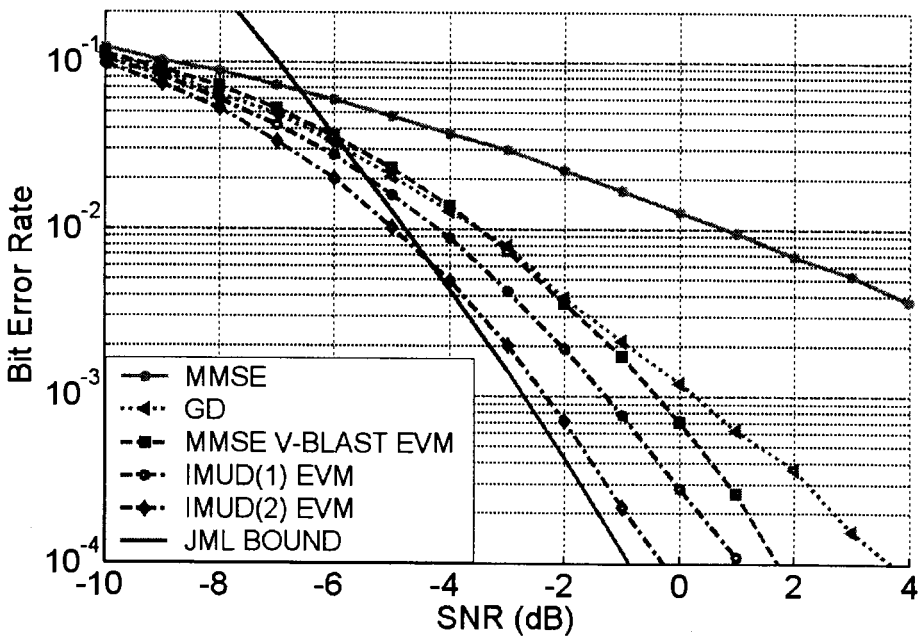


Figure 20: (12,4,3) system

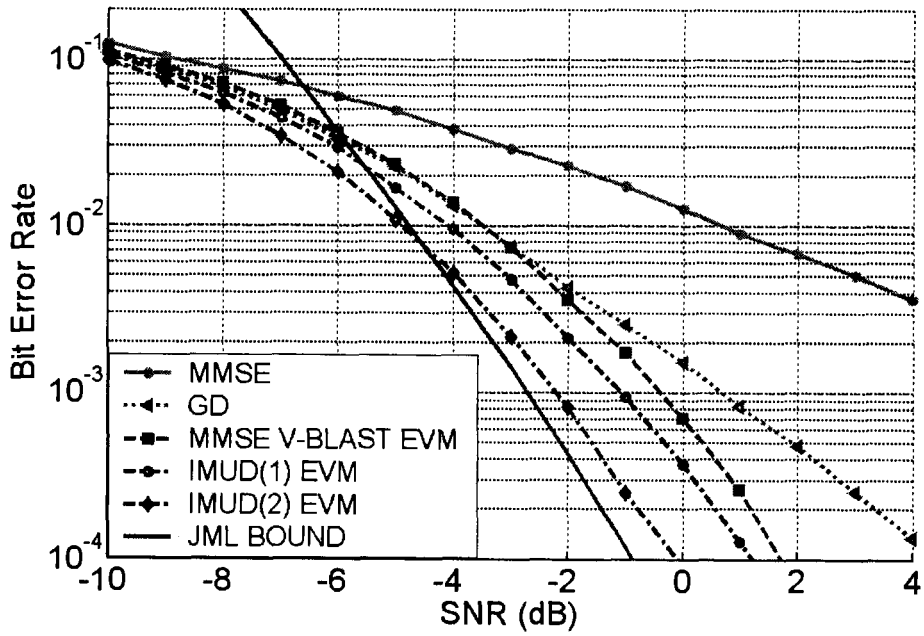
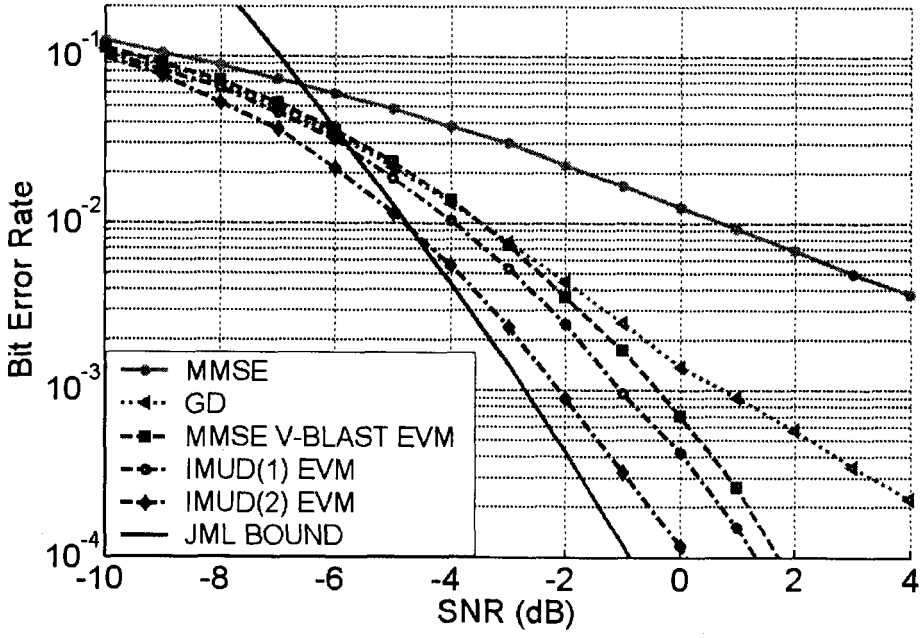



Figure 21: (12,6,2) system

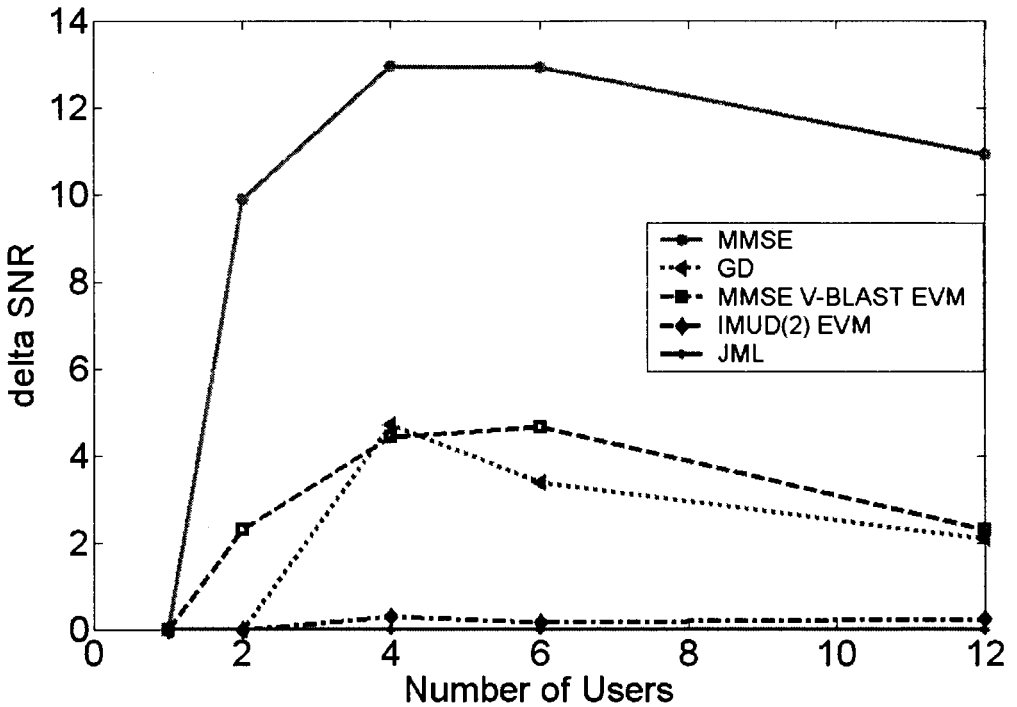


As noted in [11], the performance of the group techniques increases as the group size increases. However, the gain itself is never commented on in [11]; the performance shift is monotonic, but not necessarily consistent with the size of the group. For IMUD, the shift in the BER curve as the group size increases is not as great as for GD, but the overall performance is much closer to JML. For a BER of $10^{-3}$, IMUD shifts less than 1 dB while GD shifts around 2 dB. This difference is magnified by the diversity present in the systems, since each technique has a different BER curve slope. This diversity difference is hypothesized to be a product of the soft interference cancellation. The soft-interference cancellation benefits are discussed in Section 3.8.3.

For JML, the diversity order of any system is equal to $M$ [4]. For the $M = 12$ systems, the full diversity of the JML system is not achieved until the lower BER rates are achieved, but the increasing slope of the BER curve is observable. For the BER curves above, the diversity level of IMUD improves for larger groups. With the largest group size of 6, IMUD's diversity is nearly equal to that of JML. This comparison is done based on the slope of the BER curves, and further high SNR simulations should be completed to verify the diversity level.

As a comparison of capacity for each technique, a plot of users versus SNR was compiled for the various techniques at a BER of $10^{-3}$ in Figure 24. For IMUD and GD, the systems are (**2,1,2**), (**4,2,2**), (**6,2,3**), and (**12,2,6**). Delta SNR is the difference between the SNR of JML and the given technique.

**Figure 22: SNR penalty versus number of users at BER of $10^{-3}$ with M=N**




This figure is best examined from a design perspective; it depicts the extra SNR necessary for a technique to match the performance of JML for a given number of users. The linear technique MMSE requires the most SNR boost to match JML. Just like in the performance case where the group techniques are upper-bounded by the linear technique, it can be viewed here as a penalty bound. In this case, the linear technique shows the maximum excess SNR necessary to provide similar performance to JML. GD and MMSE V-BLAST with EVM sorting follow each other closely. IMUD(2) with EVM sorting is the closest to JML, with the SNR difference being slightly less than 0.3 dB.

### 3.8.2 IMUD with More Users than Receivers (M < N)

In overloaded systems where there are more users than receivers, the linear techniques fail. In ZF MUD, the covariance matrix is not invertible because of a

singularity. MMSE MUD counteracts the singularity with the addition of the noise power, but its performance is still severely affected. Figure 23 shows a (**4,3,2**) system and Figures 24 and 25 show (**8,6,2**) and (**8,2,6**) systems, respectively. For JML in an overloaded system, it is known [4] that the diversity level should stay the same, with a shift in the BER curve. For the group techniques, it is expected that they will be adversely affected along the same lines as the linear techniques, since they make use of linear suppression.

The overloaded performance of IMUD is excellent compared to the other techniques that use linear suppression. Comparing JML to IMUD in a (**8,2,6**) configuration comes up with startling numbers; at a BER of $10^{-3}$, IMUD comes within 1dB of JML and with less complexity (Figure 16). Moreover, IMUD provides soft output, unlike JML. With smaller groups, IMUD is shifted and loses diversity, but still manages to outperform the other techniques.

**Figure 23: (4,3,2) system**

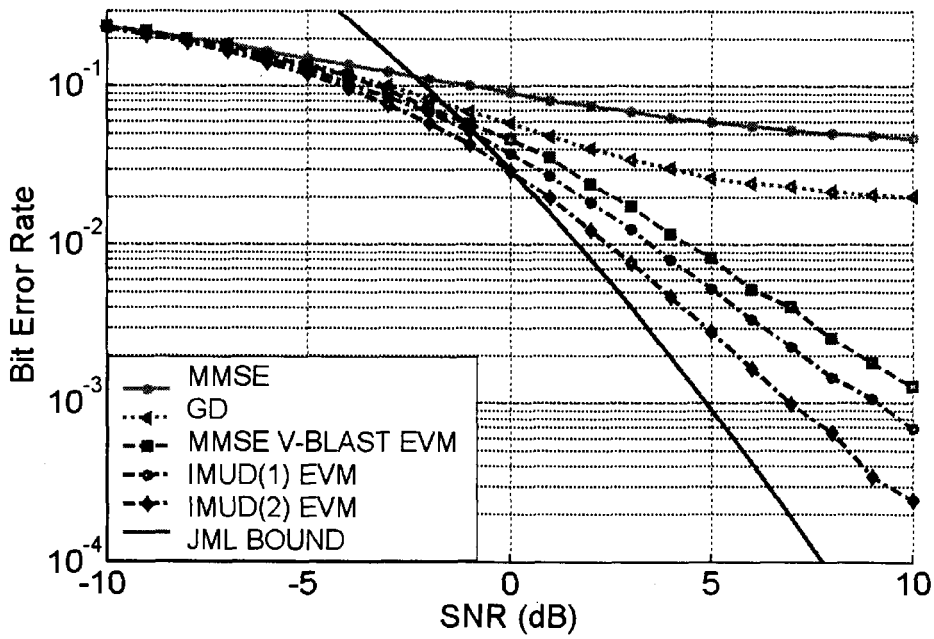


**Figure 24: (8,6,2) system**

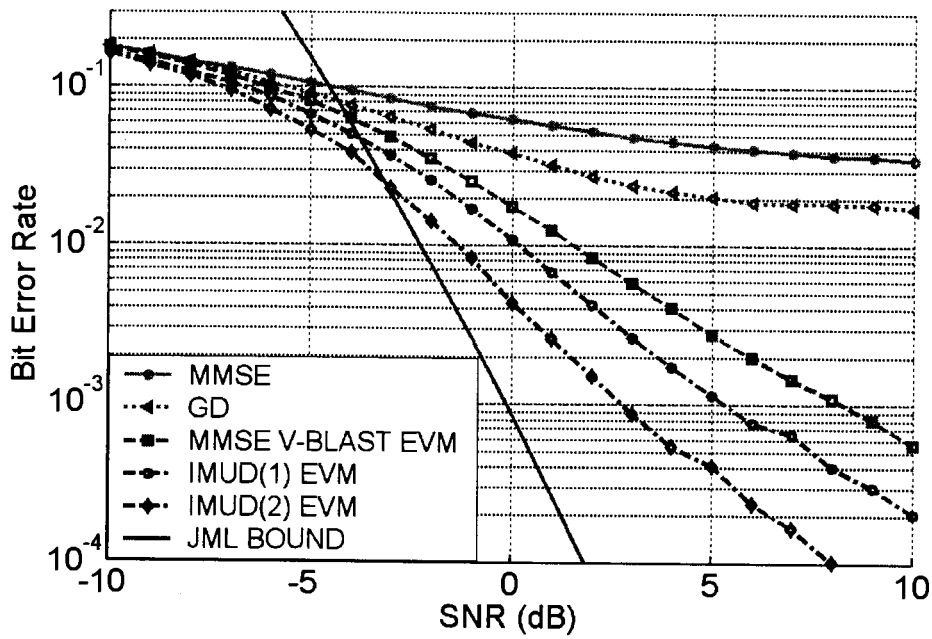
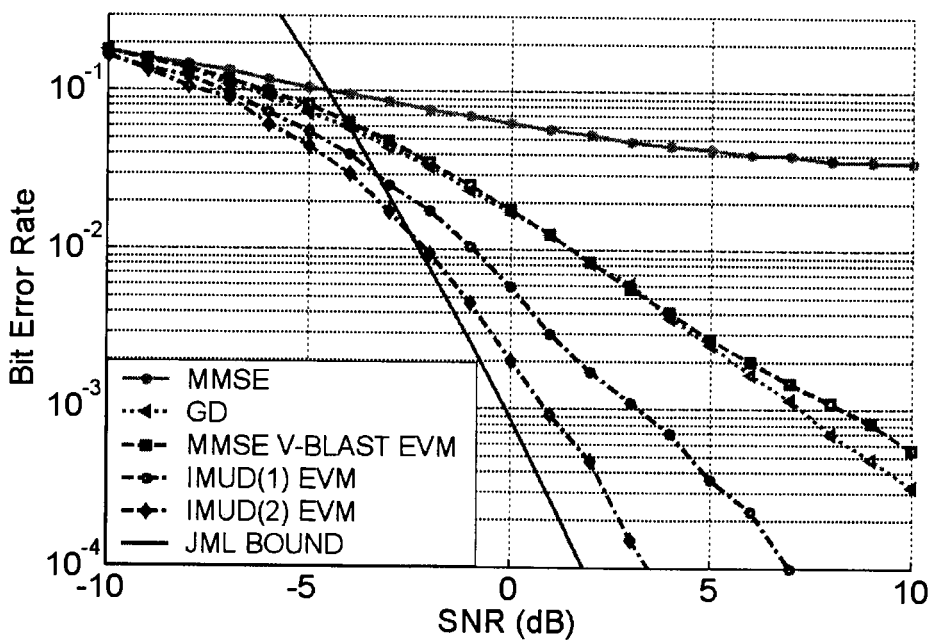


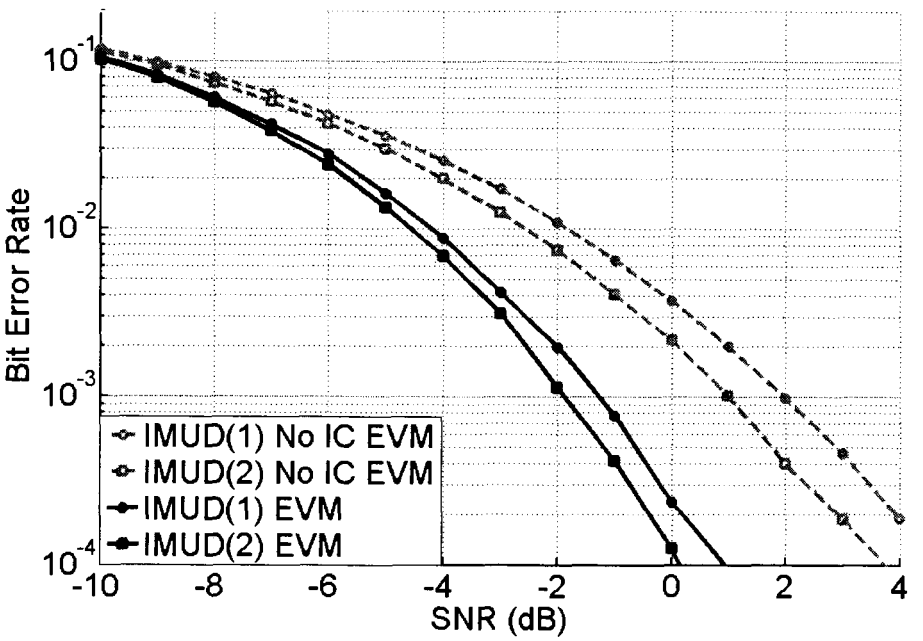

**Figure 25: (8,2,6) system**



### 3.8.3 Interference Cancellation

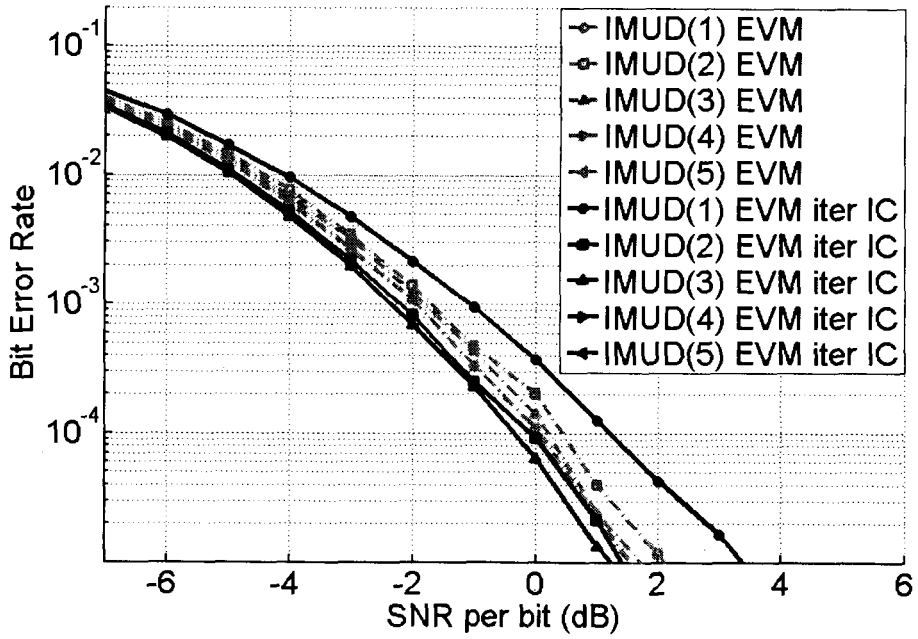
The first component of IMUD investigated was the benefit of the soft interference cancellation. In IMUD, IC is used between groups and between iterations. Initial investigations focused on IC between groups within an iteration (not between iterations) and found that it increased the slope of the BER curve, effectively increasing the diversity of the system (Figure 26). The IMUD(*iter*) no IC EVM curves refer to IMUD with no IC between groups or iterations. This result was somewhat expected, since in other systems with hard IC (V-BLAST), it is the IC that provides the performance gain over the basic linear technique. In overloaded systems, where there are more users or data streams than receive antennas ( $M < N$ ) the IC also lowered the error floor (Figures 24 and 25).

**Figure 26: (12,3,4) system with and without IC between group, no IC between iterations**



IC between iterations as well as between groups demonstrated that while the IC between iterations does not increase the BER performance of IMUD, it does allow faster convergence. The plot in Figure 27 shows that with using only IC between groups, IMUD converges in about 4 iterations; with IC between groups and iterations, IMUD converges in 2 iterations. This can easily be explained by the fact that the soft information available to the second iteration removes the known interference from all other users not in the current group. Then, the GAPP technique makes a decision with a greatly reduced interference level starting from the beginning of the second iteration by using $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\Sigma}^{(1)}$. This is unlike the first iteration where only the groups detected in that iteration are removed. The information in $\boldsymbol{\mu}^{(\gamma)}$ and $\boldsymbol{\Sigma}^{(\gamma)}$ is available to all subsequent iterations, with and without IC, in the form of the *a prioris*. However, without IC between iterations, this information is not used to its maximum benefit. The removal of the known means and variances allow the technique to converge at a faster rate (Figure 27).

**Figure 27: (12,4,3) system with and without IC between iterations**



### 3.8.4 Ordering Techniques

The contribution of user detection order to the performance of IMUD is discussed next. The V-BLAST technique in [8] uses an ordering technique based on post-filtering SNR. It has since been shown that other ordering techniques allow for better performance. One such technique is the log-likelihood ratio (LLR) [10], which bases reliability on the magnitude of the LLR of the soft decisions. For IMUD, the following ordering criteria were tested for the first iteration: random ordering, SNR, LLR, and error variance minimization (EVM). It should be remembered that IMUD uses random ordering after the first iteration. Justification for the use of randomization of user order after the first iteration is shown in Figure 29.

The post-filtering SNR sort is taken from [8]. Since only the SNR is of concern (not including the interference), a ZF filter (19) is used to suppress the other users. It should be noted that in the case of overloaded arrays, this technique will not work, since the singular matrix $\mathbf{H}^{\dagger}\mathbf{H}$ will be non-invertible. The SNR of the $i^{th}$ user is calculated from $\mathbf{W}^{\dagger}\mathbf{y}$ as

$$\mathbf{w}_i^{\dagger}\mathbf{y} = b_i + \mathbf{w}_i^{\dagger}\mathbf{n}$$

$$\text{then the SNR is } \frac{\mathrm{E}\left[b_i b_i^{*}\right]}{\mathrm{E}\left[\mathbf{w}_i^{\dagger}\mathbf{n}\mathbf{n}^{\dagger}\mathbf{w}_i\right]} = \frac{1}{N_0\left\|\mathbf{w}_i^{\dagger}\mathbf{w}_i\right\|} \tag{75}$$

Since $N_0$ is the same for all users, the user with the highest SNR is the one with the highest value for (75). The ordering technique is to find the users with the lowest value for $\left\|\mathbf{w}_i^{\dagger}\mathbf{w}_i\right\|$.

The LLR sort is from [10]. The method used with IMUD is actually a simplified version, referred to as envelope-based ordering. The LLR, introduced in (34), is derived in [10] for BPSK and MPSK systems. Like SNR, it uses linear suppression to derive the

metric. It is therefore subject to the same problem of a singular matrix in overloaded cases. For BPSK, the LLR is

$$\Lambda_{ij} = ln\left(\frac{Pr\left(b_{ij}=+1 \mid x_{ij}\right)}{Pr\left(b_{ij}=-1 \mid x_{ij}\right)}\right) \tag{76}$$

where $x_{ij} = \left(\mathbf{H}^{\dagger}\mathbf{H}\right)^{-1}\mathbf{h}_{ij}\mathbf{y} = \mathbf{w}_{ij}^{\dagger}\mathbf{y}$ is the ZF filtered sample for the $i^{th}$ user of the $j^{th}$ group. The LLR magnitude is all that is needed, since it represents the certainty of the decision (high certainty of +1 is $+\infty$, high certainty of -1 is $-\infty$). The decision itself is not important. Using $x_{ij}$ and the channel model, the magnitude of the LLR becomes

$$\left|\Lambda_{ij}\right| = \frac{4\sqrt{E_S}\left|Re\left\{x_{ij}\right\}\right|}{N_0\left\|\mathbf{w}_{ij}\right\|^2} \tag{77}$$

As pointed out in [10], (77) has some similarity to (75). The expected value of (77) is found to be $\frac{4E_S}{N_0\left\|\mathbf{w}_{ij}\right\|^2}$, which is just a scaled version of (75). The difference between the LLR sort and the SNR sort is that the LLR includes the instantaneous measurement, giving more information about the current channel state. The metric for the LLR sort can be simplified by removing the constant terms.

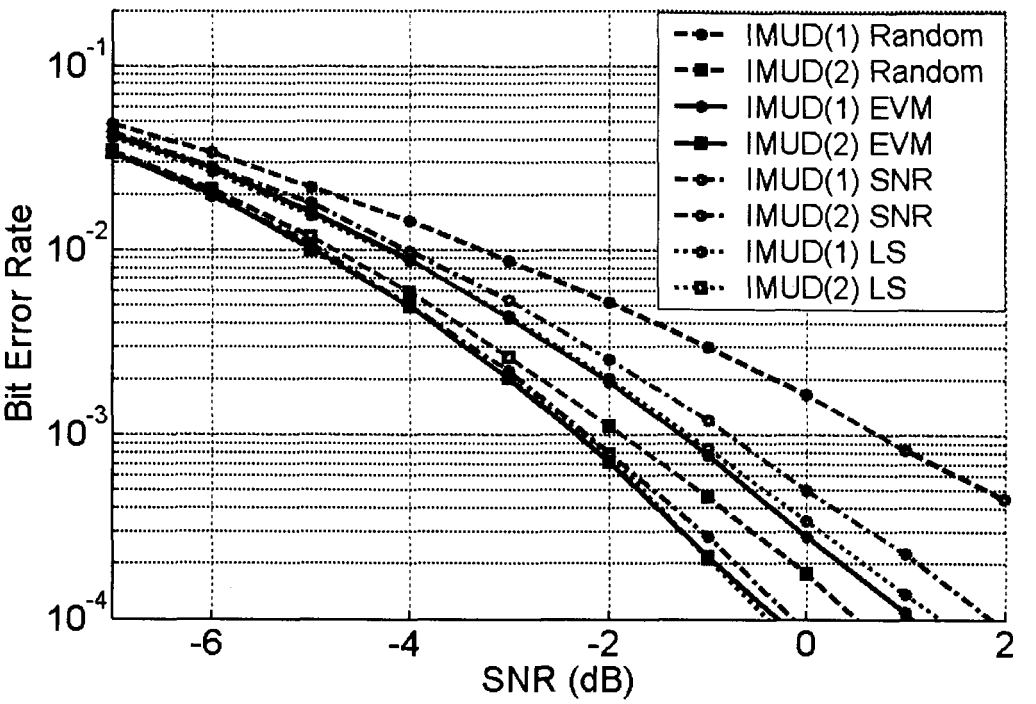
$$\text{LLR metric} = \frac{\left|Re\left\{x_{ij}\right\}\right|}{\left\|\mathbf{w}_{ij}\right\|^2} \tag{78}$$

EVM is covered in Section 3.3, with the metric being derived in (48). The ordering is based on the users with the lowest value for the mean square error after the suppression of other users. To accommodate previously detected users, (48) is modified in (51).

The results for each technique in a (**12,3,4**) system are shown in Figure 27. As detailed in Section 3.6, the second iteration for all techniques is based on a randomized order.

The random order is the worst performer, as expected. However, even with random ordering, IMUD(2) performs with a BER of $10^{-3}$ at an SNR of -2dB, which could be acceptable in many systems. In the case that the complexity of the sort algorithm is seen as excessive, Figure 28 gives evidence that even with random ordering for the first iteration, IMUD has decent performance. The SNR sort is a big improvement in both the first and second iterations. In fact, it is interesting to note that the difference at a BER of $10^{-3}$ for SNR, LLR and EVM sort is a fraction of a dB. There is a difference in the effective diversity however, since the slopes of the curves are different. The difference between LLR and EVM is minimal. With all the techniques, the spread for the $2^{nd}$ iteration is less than that after the $1^{st}$ iteration.

**Figure 28: (12,3,4) system comparison of ordering techniques**



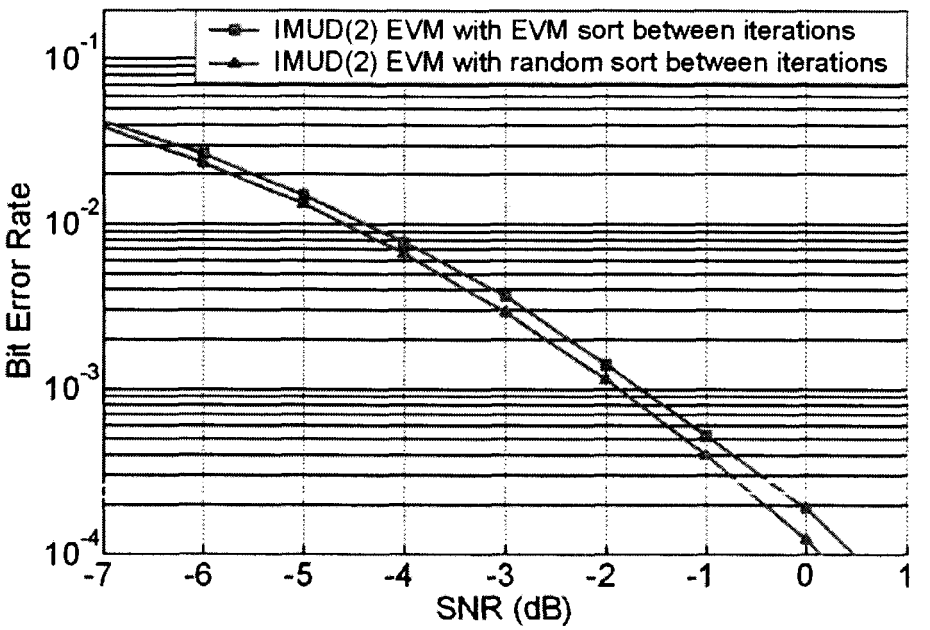

Another aspect of ordering investigated was that of ordering between iterations. The reasoning behind the iterative procedure in IMUD is that between each iteration, the groups are changed such that the correlations between users are altered. This allows the dependencies generated in one iteration to be exploited by the subsequent iteration with the transfer of the *a prioris*. To do this, IMUD is designed to use the best performing initial detection order (EVM), and randomly order the users in all subsequent iterations.

However, the EVM ordering can be modified to accept *a priori* information, specifically the variance of the user symbols. This updated EVM was tested as a replacement for the random sort between iterations in hopes that it would deliver better results. In (48), if we replace $\mathrm{E}\left[\mathbf{b}\mathbf{b}^{\dagger}\right]$ with the variance matrix $\boldsymbol{\Sigma}^{(\gamma)}$ (56) for the $\gamma^{\text{th}}$ iteration instead of using the transmitted symbol variance estimate of $\sigma_b^2\mathbf{I}$, it becomes

$$E\left[\mathbf{e}\mathbf{e}^{\dagger}\right] = \boldsymbol{\Sigma} - \mathbf{H}^{\dagger}\mathbf{R}_{yy}^{-1}\mathbf{H}\boldsymbol{\Sigma} - \boldsymbol{\Sigma}^{\dagger}\mathbf{H}^{\dagger}\mathbf{R}_{yy}^{-1}\mathbf{H} + \mathbf{H}^{\dagger}\mathbf{R}_{yy}^{-1}\mathbf{H} \tag{79}$$

where $\mathbf{R}_{yy}$ is $\mathbf{R}_{U}$ from (57). This reordering after the first iteration is compared to that of a random reordering in Figure 29.

**Figure 29: (12,3,4) system comparing random sort to EVM sort between iterations**




The figure shows that the performance is actually degraded when using the updated EVM sort. A plausible reason could include the fact that the user orderings between the first and second iteration are too correlated.

### 3.8.5 Information Transfer between Iterations

In IMUD, the APPs are passed from iteration to iteration to facilitate the evolution of the symbol probabilities. In initial investigations, it was hypothesized that the proper feedback information would be the extrinsic APPs. The extrinsic information is that

which is gained from the detection and decoding process. It is essentially the APPs with all *a priori* information removed [5].

In coded systems (Turbo codes), the channel information is also removed since it can be easily separated from the extrinsic probabilities. The channel information is a product of the detection algorithm, excluding the decoder; usually, the channel information is simply the likelihood metric from (8). The results from the detection algorithm only serve as a guide for the decoder; once the decoder has developed its output, the channel information can be removed to prevent any feedback. Observe that for any subsequent iteration, the channel information will be identical to the first since the detector itself will be using the same channel measurements. The extrinsic information is thus dependent only on the code.

However, in IMUD, the channel information is MMSE filtered, and is integral to the iterative process. In fact, the channel information is the extrinsic information since there is no other source to account for the evolving APPs. From now on, APPs refer to output LLRs $\mathbf{L}_{IMUD}$ and *a prioris* refer to input LLRs $\mathbf{L}_A$.

The expression for the APPs at the output of the $J^{th}$ iteration of IMUD is

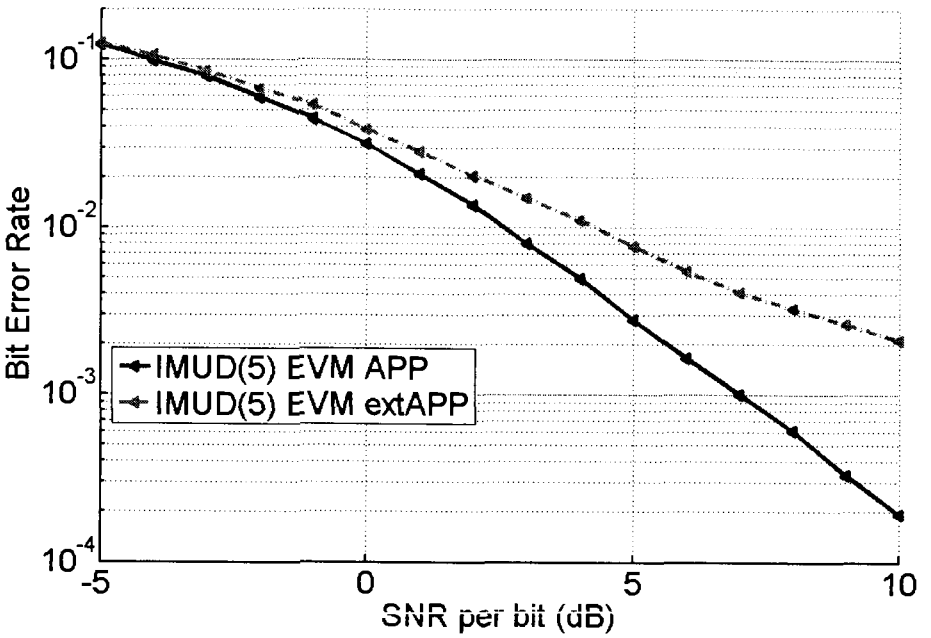
$$\begin{aligned}\mathbf{L}_{IMUD}(\gamma) &= \mathbf{L}_A(\gamma) + \mathbf{L}_{chan}(\gamma) \\ &= \mathbf{L}_A(\gamma) + \mathbf{L}_{ext}(\gamma)\end{aligned} \tag{80}$$

Using extrinsic APP feedback, $\mathbf{L}_A(\gamma+1)$ would be simply $\mathbf{L}_{ext}(\gamma)$. Using full APP feeback, $\mathbf{L}_A(\gamma+1)$ would be $\mathbf{L}_A(\gamma)+\mathbf{L}_{ext}(\gamma)$.

IMUD with no IC between iterations and with extrinsic APP feed-back exhibited an error floor. The fact that the error floor occurs after the second iteration points to the problem in the extrinsic APPs. For the first iteration, the symbols are equiprobable, and thus the APP using either the extrinsic or full APPs is simply $\mathbf{L}_{ext}(1)$. However, for the

second iteration, the APPs differ. Figure 29 shows the performance in a (**4,3,2**) system for IMUD(5) with no IC between iterations with both feedback probabilities.

**Figure 30: (4,3,2) system with no IC between iterations**




This is the reverse of what is expected based on the use of extrinsic APPs in Turbo codes. A hypothesis to explain this phenomenon is as follows. Most of the knowledge of the user symbols is extracted in the first iteration. For the next iteration (iterations are still denoted with $\gamma$), $\mathbf{L}_{ext}(\gamma+1)$ will be almost equal to $\mathbf{L}_{A}(\gamma+1)$. This is because the *a priori* knowledge $\mathbf{L}_{A}(\gamma+1)=\mathbf{L}_{ext}(\gamma)$. This may cause a positive feedback, thus creating an error floor. The error floor only occurs after the correlation between the LLRs becomes high enough.

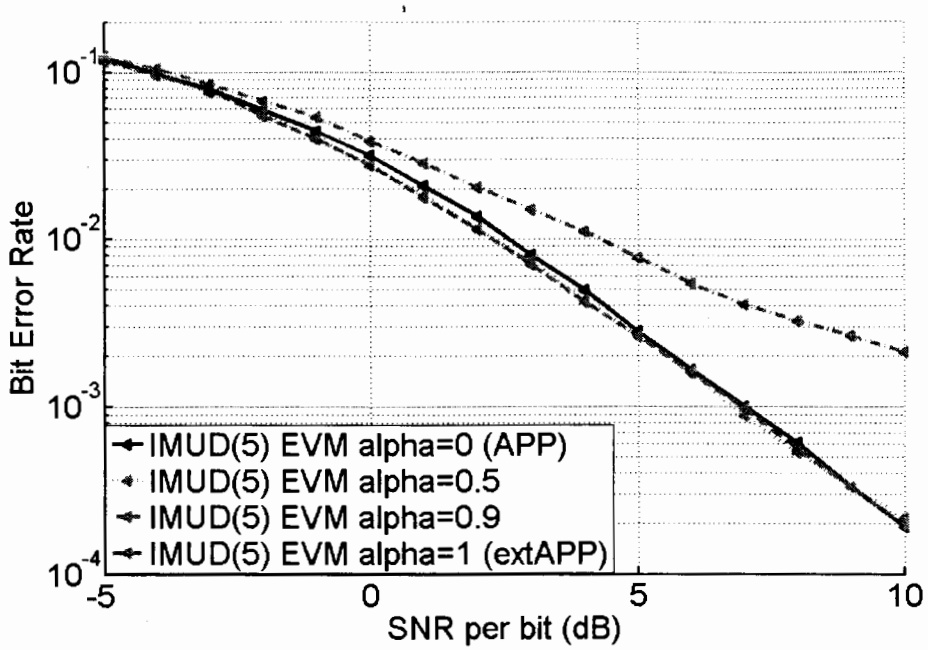
In Turbo codes, the problem is the reverse. The output APP LLR is defined as $\mathbf{L}_{APP}$. The feedback problem occurs when using the full APP, $\mathbf{L}_{APP}(\gamma)=\mathbf{L}_{A}(\gamma)+\mathbf{L}_{chan}(\gamma)+\mathbf{L}_{ext}(\gamma)$. The second iteration results in

$\mathbf{L}_A(2) = \mathbf{L}_{chan}(1) + \mathbf{L}_{ext}(1)$ due to equiprobable symbols. Continuing this process, the third iteration results in $\mathbf{L}_A(3) = (\mathbf{L}_{chan}(1) + \mathbf{L}_{ext}(1)) + (\mathbf{L}_{chan}(2) + \mathbf{L}_{ext}(2))$. $\mathbf{L}_{ext}$ will evolve over time to a steady-state, but the $\mathbf{L}_{chan}$ will not since it is straight from the channel measurements. Since $\mathbf{L}_{chan}$ steadily accumulates, an error floor results.

The solution is to limit the APP somehow. Most of the literature in Turbo codes solves this problem by removing all of the known information and feeding back only the extrinsic. In this case, the second iteration $\mathbf{L}_A$ is simply $\mathbf{L}_{ext}$ from the last, so $\mathbf{L}_A(2) = \mathbf{L}_{ext}(1)$. Now, when the *a priori* and channel LLR is removed from $\mathbf{L}_{APP}(2)$, we are only left with $\mathbf{L}_A(3) = \mathbf{L}_{ext}(2)$. Another way that this correlation or positive feedback can be disrupted is to use a weighted removal of the APPs [18]. The weighting factor is denoted by $\alpha$. For Turbo codes, $\mathbf{L}_A(\gamma) = \mathbf{L}_{APP}(\gamma - 1) - \alpha(\mathbf{L}_{chan}(\gamma - 1) + \mathbf{L}_A(\gamma - 1))$. In the above discussion, using the APPs as *a priori*s is the same as setting $\alpha = 0$, and using the extrinsic APPs is the same as setting $\alpha = 1$. In [18], it is shown that for the method of updating the *a prioris*, the optimum value of $\alpha$ is 1, corresponding to the feedback of only the extrinsic APPs.

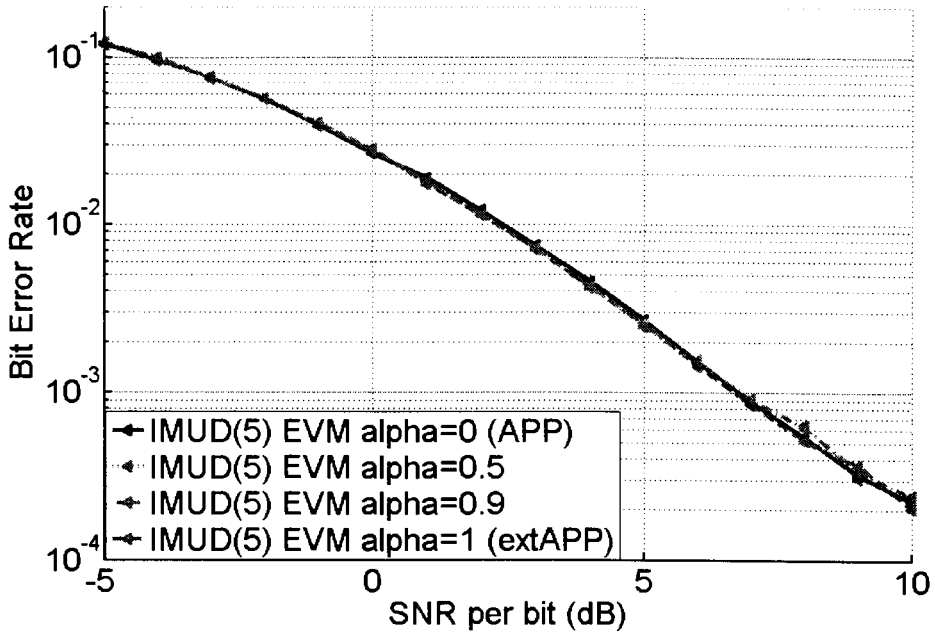
The possibility of a different optimum value of $\alpha$ for IMUD was explored next. The expression for the *a priori* is $\mathbf{L}_A(\gamma) = \mathbf{L}_{APP}(\gamma - 1) - \alpha(\mathbf{L}_A(\gamma - 1))$. Figure 30 shows the results for $\alpha$ of 0, 0.5, 0.9 and 1 for without IC between iterations. Even with only 10% of the previous *a prioris* included in the next iteration, the error floor is eliminated. This seems to point to the idea that the correlation between the *a prioris* and the extrinsic information is easily broken.

**Figure 31: (4,3,2) system no IC between iterations**




The above discussion of information transfer between iterations considered the case of no IC between iterations. However, in IMUD, soft IC is used between iterations. This option is not usually covered in Turbo implementation. However, Sellathurai and Haykin [16] make use of it in a SISO MRC MUD concatenated with a SISO convolutional decoder. In IMUD's case, it is hypothesized that the removal of the mean values changes the measurement vector enough so that $\mathbf{L}_{ext}(1)$ and $\mathbf{L}_{ext}(2)$ are generated under much different conditions. Thus, they are uncorrelated enough to allow convergence with any value of $\alpha$. Figure 31 extends the results from Figure 30 to include IC between iterations. It turns out that $\alpha$ changes nothing in this system; the IC is enough.

**Figure 32: (4,3,2) system, IC between Iterations**




### 3.8.6 Summary

As demonstrated in simulations with results in Figures 18 to 21, IMUD in a system of equal numbers of users and receivers is capable of performance within 1dB of the optimum MUD technique with less complexity (Section 3.7). There are further possible methods to reduce complexity of IMUD as well, including more efficient ordering techniques and matrix inverse updates.

Figures 24 and 25 also show that in the overloaded case, IMUD outperforms MMSE V-BLAST, which is seen as the closest competitor. Complexity in these systems is similar to the equal user to receiver case.

# 4 ITERATIVE MULTIUSER DETECTION IN A CONCATENATED SYSTEM

IMUD is capable of being a stage in a concatenated system as well as a stand-alone multiuser detector. This is due to IMUD's soft input and output. One realization is a two-stage serially concatenated system, where IMUD is the initial multiuser detector at the receiver. In this case, the transmitter consists of one non-systematic convolutional encoder per user or stream, followed by one interleaver for each encoder. The receiver consists of the IMUD detector followed by the respective de-interleavers for each user or stream, then the respective SISO decoders. If the multiuser channel were replaced by a channel with delay spread, the detector becomes a SISO equalizer; this is the much studied Turbo Equalizer [14].

The following sections 4.1 – 4.3 lay out the IMUD/SISO concatenated system in detail. Results are reported in Section 4.4. The notation for the IMUD/SISO concatenated system is an extension of what was used before. Time is now relevant for the SISO block, since the probabilities at the output are based on the encoder memory, unlike the symbol-by-symbol decisions in IMUD. $u_n(t)$ is the uncoded multi-bit symbol and $c_n(t)$ is the coded multi-bit symbol for user *n* at time *t*. The number of bits in a symbol depends on the specific code that is used (Section 4.2). $L^{(\Gamma)}(c_n(t))$ refers to the LLR for symbol $c_n(t)$ for the $\Gamma^{\text{th}}$ iteration, $\mathbf{L}^{(\Gamma)}(c_n)$ refers to a vector of LLRs for symbol $c_n$ for the entire block at the $\Gamma^{\text{th}}$ iteration, and $\mathbf{L}^{(\Gamma)}(\mathbf{c})$ refers to a vector of LLRs for symbol *c* for the entire block and all users. The length of the convolutionally encoded blocks is denoted as *d*, and for the simulations $d = 500$.

## 4.1 System Description

**Figure 33: Concatenated SISO system**

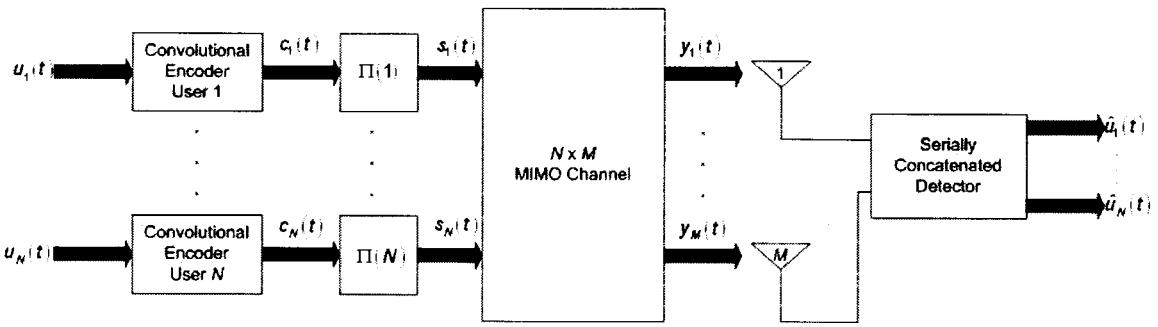
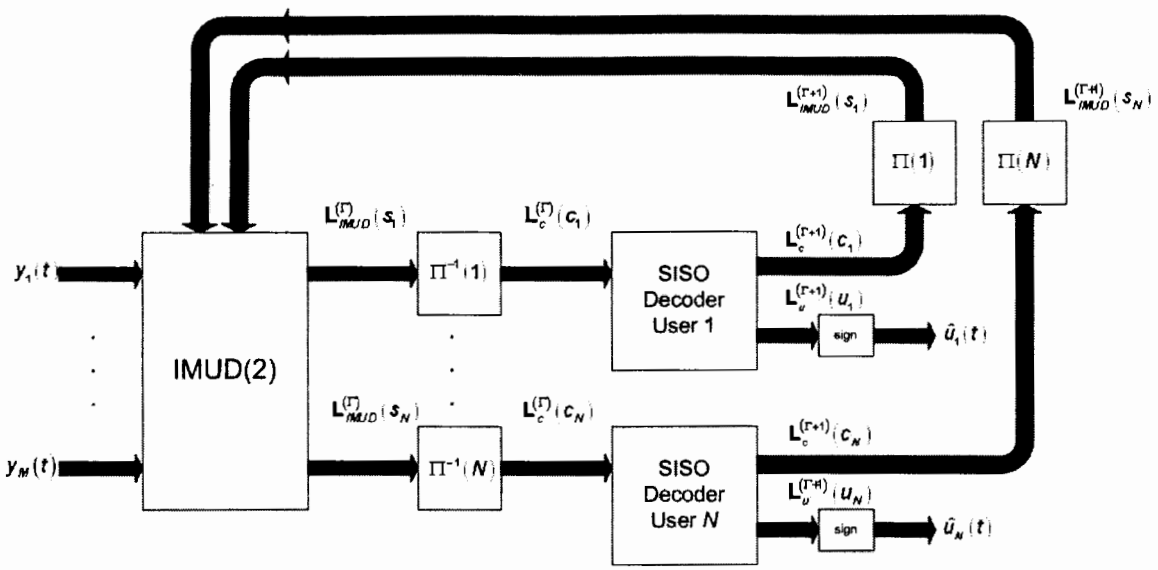



Figure 33 shows the structure of the concatenated system. For the $n^{th}$ user, the uncoded data stream $u_n(t)$ is independently encoded with a convolutional code and is output as $c_n(t)$. Each coded stream is then bit interleaved, with the interleaver denoted as $\Pi(n)$. The output of the interleaver is $s_n(t)$. The code can be the same for each user if each user has a unique interleaver. Each interleaver does two tasks. First, it breaks up channel fades that last longer than the constraint length of the code. Second, the different interleavers also add insurance for the MUD in the case of mutually correlated channels. By making the fades for each user independent, if one user experiences a fade during a symbol, the other users may not. MUD depends on users with good channels to aid in detection of users with bad channels. This is an important issue, since the last detected users in IMUD are normally in a faded channel, and are dependent on the correct decisions of the initially detected users.

The interleaved and coded bits $s_n(t)$ are then transmitted over the MIMO channel and received by the concatenated detector as $y_m(t)$. The detector is explained in Figure 34.

**Figure 34: Concatenated detector**




The detector is made up of the IMUD block running two iterations, and $N$ interleavers, de-interleavers and SISO decoders, one for each user or stream. The superscript $\Gamma$ is used to denote the iterations of the entire concatenated decoder. Keep in mind there are now two effective iterative processes: the IMUD iterative process, which has iterations denoted with $\gamma$, and the concatenated decoder, which has iterations denoted with $\Gamma$. However, IMUD(2) will be treated here as a black box and the inner workings of IMUD as reviewed in Section 3 will not be an issue. It will input the received samples $\mathbf{y}(t)$ and continue with symbol-by-symbol detection. The LLRs at the output of IMUD(2) are buffered until the entire block has been received. The flow of the rest of the detector is similar to other serial concatenated schemes. The coded symbol LLRs $\mathbf{L}_{IMUD}^{(\Gamma)}(s_n)$ are input to the deinterleavers. The deinterleaved LLRs $\mathbf{L}_{IMUD}^{(\Gamma)}(c_n)$ are then detected with the SISO convolutional decoder, which is defined in Section 4.3. The decoder outputs two soft decision streams; one for the coded symbols, $\mathbf{L}_c^{(\Gamma+1)}(c_n)$, and one for the uncoded symbols, $\mathbf{L}_u^{(\Gamma+1)}(u_n)$. The new LLRs for the coded symbols are

updated with the redundant information of the code. These LLRs are then fed back to the IMUD block after being re-interleaved as $\mathbf{L}_{IMUD}^{(\Gamma+1)}(s_n)$. The next iteration then has updated *a prioris* to work with. Decisions are made on $\mathbf{L}_{IMUD}^{(\Gamma+1)}(u_n)$. The notation for LLRs for all users and all time is $L_{IMUD}^{(\Gamma)}(\mathbf{u})$.

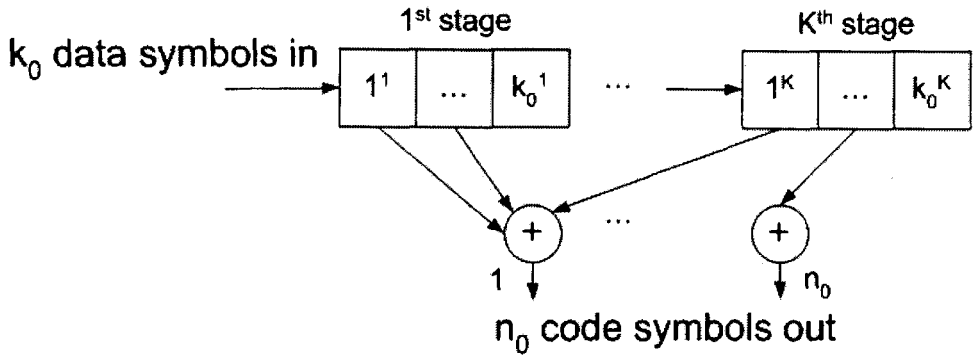
## 4.2 Encoder

Convolutional coding is a method in which redundant information is introduced into the data stream. The existing information is augmented in such a way as to allow some level of error correction. A convolutional code is traditionally implemented as a shift register with taps to modulo-2 adders. Using the notation from [1], $k_0$ bits are entered every symbol time into the shift register with $K$ serially connected stages for a total shift register length of $k_0K$ symbols. $K$ is referred to as the constraint length of the code, and is the number of cycles that a given symbol affects the output of the coder. At the output of the encoder, $n_0$ bits are shifted out every cycle. For the simulated concatenated system, the input block consists of $d$ clock cycles, or $d \cdot k_0$ input symbols. The convolutional code is left in an undetermined state instead of being returned to a known state. It is possible to have sections with varying constraint length, however the largest section sets the constraint length for the system.

The convolutional codes used in the simulated concatenated system are taken from [1]. The codes are expressed as vectors of octal numbers, known as generators. Each input path has its own vector. The octal number denotes the taps on the shift register that are summed to produce the outputs. Convolutional codes are also described with a rate, which is the number of input bits, $k_0$, for the number of output bits,

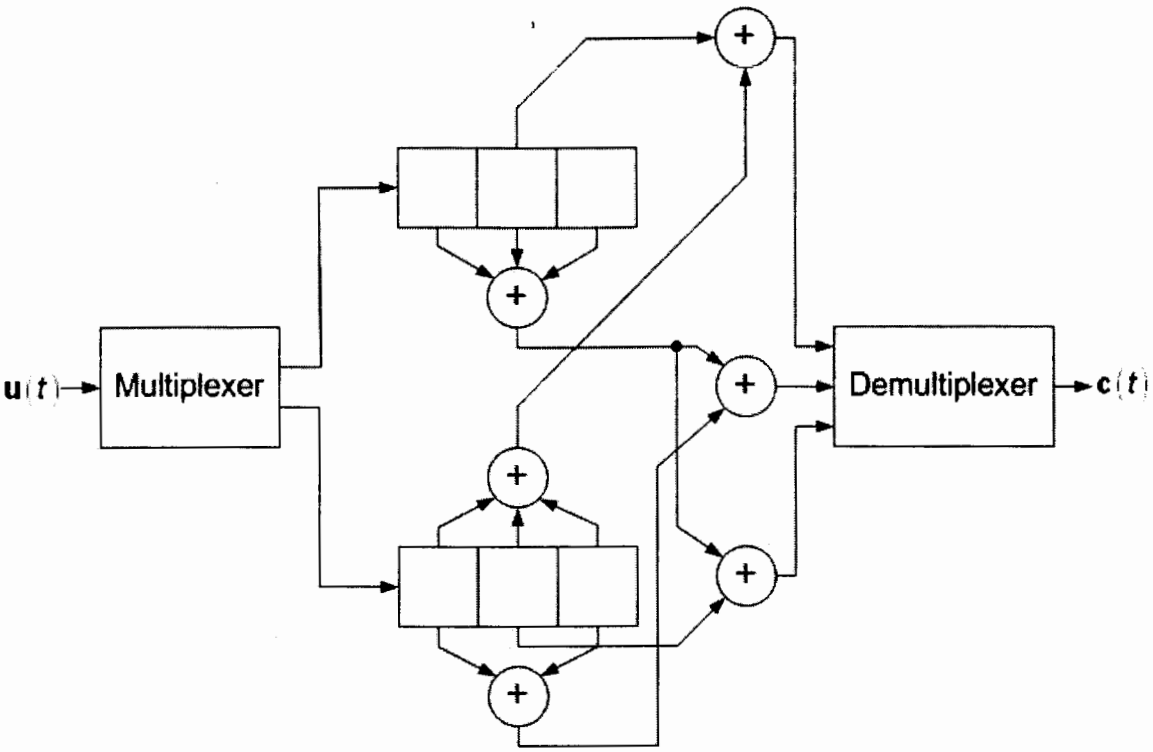
$n_0$, for a code rate of $k_0/n_0$. There are then $n_0$ generator vectors of length $k_0$. Figure 35 below shows a generic convolutional encoder.

**Figure 35: Rate $n_0/k_0$ convolutional encoder**




The octal vectors represent which locations in each stage are summed to produce a certain output bit. The octal number starts with the first input location as the LSB. In the simulations to follow, a rate 1/3 and rate 2/3 code is simulated. For the rate 2/3 code with generators (2,7), (7,5), and (7,2), $n_0 = 3$, $k_0 = 2$, and $K = 3$. The data symbols are multiplexed into two streams and input into the encoder. The output is then de-multiplexed for symbol-by-symbol BPSK transmission over the mobile channel. The encoder for the rate 2/3 code is depicted in Figure 35.

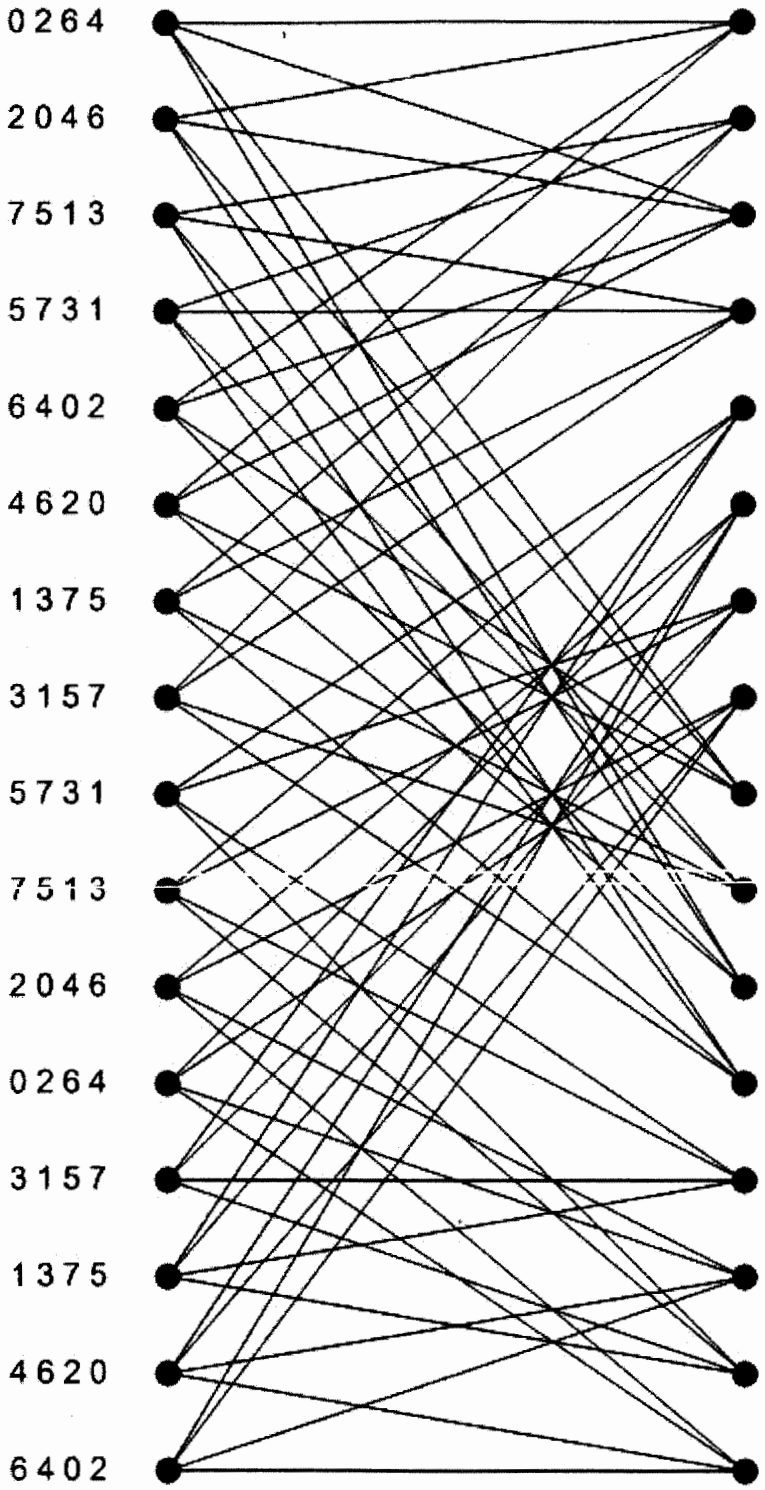
**Figure 36: Rate 2/3 non-systematic convolutional encoder**




Note how the tap positions from the generator vectors match up with Figure 36. For example, the vector for the third output is (7,2). After conversion from octal to binary, they correspond to all three memory locations from the first input summed together or (1,1,1) and a single tap from the second memory location from the second input or (0,1,0).

For systems with memory of finite length, a trellis representation can be formed. The states are defined with the constraint length $K$ and the number of input bits $k_0$. For an $m$-ary constellation, each code has a trellis with $m^{k_0(K-1)}$ states. For the rate 2/3 code above, the trellis contains $2^{2(2)} = 16$ states. The trellis is shown in Figure 37.

**Figure 37: Trellis of rate 2/3 convolutional code shown in Figure 36**



## 4.3 Decoder

**Figure 38: Non-systematic SISO decoder block**


The soft-input soft-output (SISO) block [17] is a derivative of the BCJR algorithm [12]. Both are optimum decoders for memory based systems, and can be used for equalizers or convolutional decoders. It uses the redundant information introduced by the channel or code to improve the symbol probabilities.

Adapting the notation from [17], the SISO detector is used as follows. As shown in Figure 37, the SISO block has two soft inputs and two soft outputs. The inputs are used as *a prioris* for both the coded and uncoded bits (**c** and **u**). The outputs are updated within the SISO decoder and output as probabilities.

The updated probabilities are extrinsic APPs. Following the discussion in Section 3.8.5 about APPs, the output LLRs for the coded bits will be examined. They are

$$\begin{aligned} \mathbf{L}_c^{(\Gamma+1)}(\mathbf{c}) &= \mathbf{L}_A(\mathbf{c}) + \mathbf{L}_{ext}(\mathbf{c}) \\ &= \mathbf{L}_c^{(\Gamma)}(\mathbf{c}) + \mathbf{L}_{ext}(\mathbf{c}) \\ &= \left(\mathbf{L}_A^{(\Gamma)}(\gamma = 2) + \mathbf{L}_{chan}^{(\Gamma)}(\gamma = 1)\right) + \mathbf{L}_{ext}(\mathbf{c}) \end{aligned} \tag{81}$$

(81) shows the LLR at the output of the SISO decoder as a sum of the *a priori* into the decoder (the *a prioris* into the IMUD block plus the channel LLRs developed during the 2 iteration IMUD) and the extrinsic information developed during the decoding process. Remember, the extrinsic information in IMUD is the channel information, and since

IMUD is run for 2 iterations and no *a prioris* are removed, both iterations channel measurements are included. The difference between the SISO APPs and the IMUD APPs is that the extrinsic information in the SISO block is not the channel information; in fact, the channel information is not directly included at all since the SISO block only works on existing probabilities, $L_A(\mathbf{c})$. These *a prioris* include the channel measurement left from the IMUD(2) block, as well as the redundant code information. After the SISO has produced its APPs, the *a priori* information must be removed to prevent any feedback. This is unlike the IMUD block, where they are included to weight the probabilities to avoid correlation between iterations. It should be noted that while the input LLRs for the uncoded symbols $\mathbf{L}_u^{(\Gamma)}(\mathbf{u})$ are equi-probable for every iteration of IMUD/SISO, the output of each iteration $\mathbf{L}_u^{(\Gamma+1)}(\mathbf{u})$ evolves so that it may be used for the final decision.

The SISO block is run over the entire coded symbol block, which consists of $d \cdot k_0$ input and $d \cdot n_0$ output symbols. As with IMUD, the calculations within the SISO block will be done with the probabilities and not the LLR's.

Since our system consists of a code that is of rate $k_0/n_0$, each code trellis branch corresponds to an input symbol consisting of $n_0$ bits and an output symbol consisting of $k_0$ bits, shown in Figure 37. Since the bits in these symbols are co-dependent because of the code trellis branches, the *a priori* notation is changed to $\Pr_{in}(\mathbf{c}^{\eta}(t))$ and $\Pr_{in}(\mathbf{u}^{\kappa}(t))$, where *in* refers to the fact that the probabilities are *a priori*, $\eta$ references the coded bits $1 \ldots n_0$, and $\kappa$ references the data bits $1 \ldots k_0$. $t$ runs from 1 to $d$, and references the clock cycle.

Because of interleaving, the coded bits on each branch are assumed to have undergone independent fades, and thus have independent input probabilities. For user $n$, the input probability for an entire branch at time $t$ is then

$\Pr_{in}\left(c_n(t)=C\right)=\prod_{\eta=1}^{n_0}\Pr_{in}\left(c_n^{\eta}(t)=c^{\eta}\right)$ where $C=\left(c^1|\cdots|c^{n_0}\right)$ and

$\Pr_{in}\left(u_n(t)=U\right)=\prod_{\kappa=1}^{k_0}\Pr_{in}\left(u_n^{\kappa}(t)=u^{\kappa}\right)$ where $U=\left(u^1|\cdots|u^{k_0}\right)$ refer to $n_0$ and $k_0$ bit length symbols, respectively. In the following equations, $s_t$ refers to the trellis state at time $t$, and the variables $\alpha$ and $\beta$ are from the BCJR algorithm [12].

Using the technique from [11], the output probabilities for the coded bits are

$$Pr_{out}\left(c_n^{\eta}(t)=c\right)=H_{c_n^{\eta}}\cdot Pr'_{out}\left(c_n^{\eta}(t)=c\right)$$
$$H_{c_n^{\eta}}\sum_{\left(c_n^{\eta}=c\right)\subset\left(s_{t-1}\to s_t\right)}\alpha\left(s_{t-1}\right)Pr_{in}\left(u_n(t)=U\right)\left(\prod_{\substack{i=1\\ i\neq\eta}}^{n_0}Pr_{in}\left(c_n^{i}(t)=c^{i}\right)\right)\beta\left(s_t\right) \tag{82}$$

and the output probabilities for the uncoded bits are

$$Pr_{out}\left(u_n^{\kappa}(t)=u\right)=H_{u_n^{\kappa}}\cdot Pr'_{out}\left(u_n^{\kappa}(t)=u\right)$$
$$H_{u_n^{\kappa}}\sum_{\left(u_n^{\kappa}=u\right)\subset\left(s_{t-1}\to s_t\right)}\alpha\left(s_{t-1}\right)\left(\prod_{\substack{i=1\\ i\neq\kappa}}^{k_0}Pr_{in}\left(u_n^{i}(t)=u^{i}\right)\right)Pr_{in}\left(c_n(t)=C\right)\beta\left(s_t\right) \tag{83}$$

The symbols $c^i$ and $u^i$ are dependent on the current branch being enumerated. Note, for example, that $c^{\eta}=c$, and is not included in the product of *a prioris* in (82). $H_{c_n^{\eta}}$ and $H_{u_n^{\kappa}}$ are normalization values that are defined as

$$H_{c_n^{\eta}}=\frac{Pr'_{out}\left(c_n^{\eta}(t)=c\right)}{\sum_{c}Pr'_{out}\left(c_n^{\eta}(t)=c\right)} \tag{84}$$

and

$$H_{u_n^\kappa} = \frac{Pr'_{out}\left(u_n^\kappa(t)=u\right)}{\sum_u Pr'_{out}\left(u_n^\kappa(t)=u\right)} \tag{85}$$

To make (82) and (83) more straightforward to understand, they are broken into the terms used in the BCJR. This is where the terms $\alpha(s_t)$ and $\beta(s_t)$ stem from, and are termed the forward and backward terms for time $t$ and state $s_t$, respectively. Rewriting (82) with BCJR and log terms

$$Pr_{out}\left(c_n^\eta(t)=c\right) = H_{c_n^\eta} \sum_{\left(c_n^\eta=c\right)\subset\left(s_{t-1}\to s_t\right)} e^{A(s_{t-1})+G_c^\eta(s_{t-1},s_t)+B(s_t)} \tag{86}$$

where

$$A(s_t) = log\left(\alpha(s_t)\right) = log\left(\sum_{s_{t-1}\to s_t} e^{A(s_{t-1})+G(s_{t-1},s_t)}\right) \tag{87}$$

$$B(s_t) = log\left(\beta(s_t)\right) = log\left(\sum_{s_t\to s_{t-1}} e^{B(s_{t+1})+G(s_t,s_{t-1})}\right) \tag{88}$$

$$G_c^\eta(s_{t-1},s_t) = log\left(Pr_{in}\left(u_n(t)=U\right)\left(\prod_{\substack{i=1\\ i\neq\eta}}^{n_0} Pr_{in}\left(c_n^i(t)=c^i\right)\right)\right) \tag{89}$$

$$G_u^\kappa(s_{t-1},s_t) = log\left(\left(\prod_{\substack{i=1\\ i\neq\kappa}}^{k_0} Pr_{in}\left(u_n^i(t)=u^i\right)\right) Pr_{in}\left(c_n(t)=C\right)\right) \tag{90}$$

and

$$G(s_{t-1},s_t) = log\left(Pr_{in}\left(u_n(t)=U\right) Pr_{in}\left(c_n(t)=C\right)\right) \tag{91}$$

The term $G$ is basically the likelihood ratio scaled by the *a priori* information. The reason behind having three different $G$'s is that (89) and (90) are used to form the extrinsic

APPs for **c** and **u**, while (91) is used to form *A* and *B* in (87) and (88). When programming this algorithm, it is simple to first build $G(s_{t-1}, s_t)$ for every branch for the transition then remove the probability for the specific symbol whose APP is being calculated. However, you must be sure that no rounding occurs. For example, if the *a priori* is very large (very possible for the later iterations or for high SNR), platforms like Matlab may approximate it as $\infty$. This is why the *a priori* for the $\eta^{\text{th}}$ coded bit or $\kappa^{\text{th}}$ uncoded bit is not included in the first place. It is important to remember to remove these probabilities, since if they aren't removed, the positive feedback mentioned in Section 3.8.5 will limit the system.

*A* and *B* are the forward and backward probabilities from the BCJR, calculated with *G* and the past transitions *A* and *B*. Since *A* and *B* are calculated recursively, a starting point is necessary. Since *t* refers to each symbol transmitted from 1 to *d*, *t* references the state transitions. In the same way, *t* can reference the states themselves, with the difference that the states start at $t = 0$. Since $\alpha(s_t) = \Pr(c_n(1,2,\cdots t-1), s_{t-1})$, for the starting state at $t = 1$, $\alpha(s_1) = \Pr(s_0)$. For a convolutional encoder that starts in its first state, this corresponds to $\alpha(s_1 = 1) = 1$ and $\alpha(s_1 \neq 1) = 0$, or $A(s_1 = 1) = 0$ and $A(s_1 \neq 1) = -\infty$. For the backward probabilities, the same approach can be taken where the final state is fixed. However, it would require extra transmitted symbols to insure that the trellis was in a fixed state at the end of the block. To avoid this extra complexity, equal probability can be applied to all of the states without affecting the performance of the decoder [5]. Also, scaling for *A* and *B* is arbitrary, as long as the relative levels are correct. Appropriate starting values for *B* used in the upcoming simulations are $B(s_d = s) = 0$ for all *s*.

## 4.4 Performance

The concatenated receiver of IMUD and a SISO decoder operating on a multiuser channel was first simulated with 6 users and 4 receive antennas ($N = 6$, $M = 4$) with non-systematic rate 1/3 and 2/3 codes. The rate 1/3 code has generator vectors of (4), (5), and (7) and $K = 3$, and the rate 2/3 code has generator vectors of (2,7), (7,5) and (7,2) and $K = 3$. The chosen block length was $d = 500$, which was chosen to match block sizes in existing communications standards. The larger block also allows for large interleavers, especially when compared to the constraint length of the codes. This way, fades in the channel are broken up, and error bursts are less likely. The fading rate for the system is $f_d T = 0.01$.

Figures 39 and 40 show the results for the rate 1/3 and rate 2/3 code, respectively. The notation in the graphs are $R = n_0 / k_0$ (IMUD/SISO iter, IMUD iter) $f_d T$, where iter refers to the number of iterations run for either within IMUD or within the IMUD/SISO detector. A lower bound is included in the form of a single user with the same code operating in a system with 4 receive antennas. Since the single-user case does not require a multiuser detector, it only requires one pass of the SISO decoder. It is denoted in the graphs as SU.

**Figure 39: (4,3,2) IMUD/SISO concatenated system with rate 1/3 code**

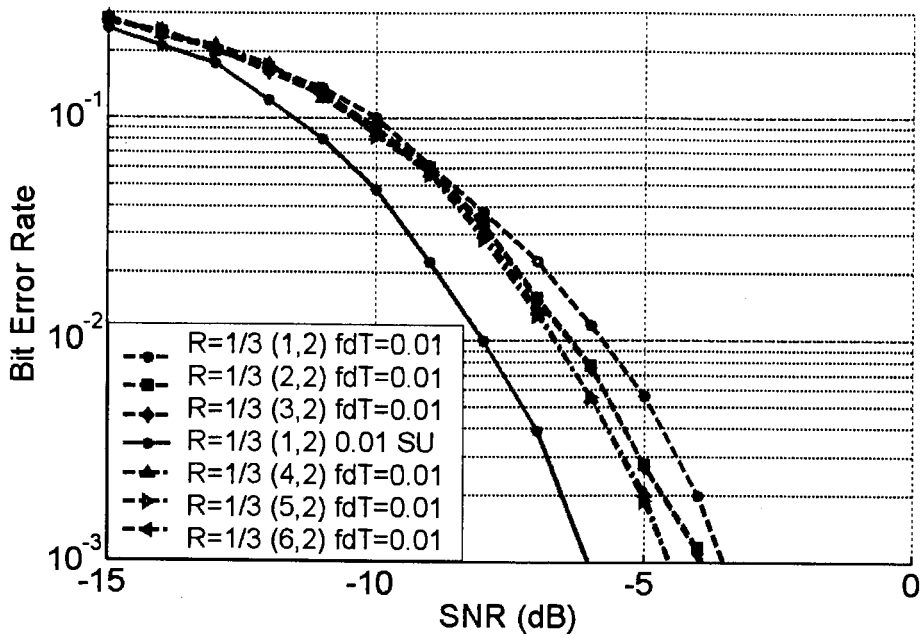
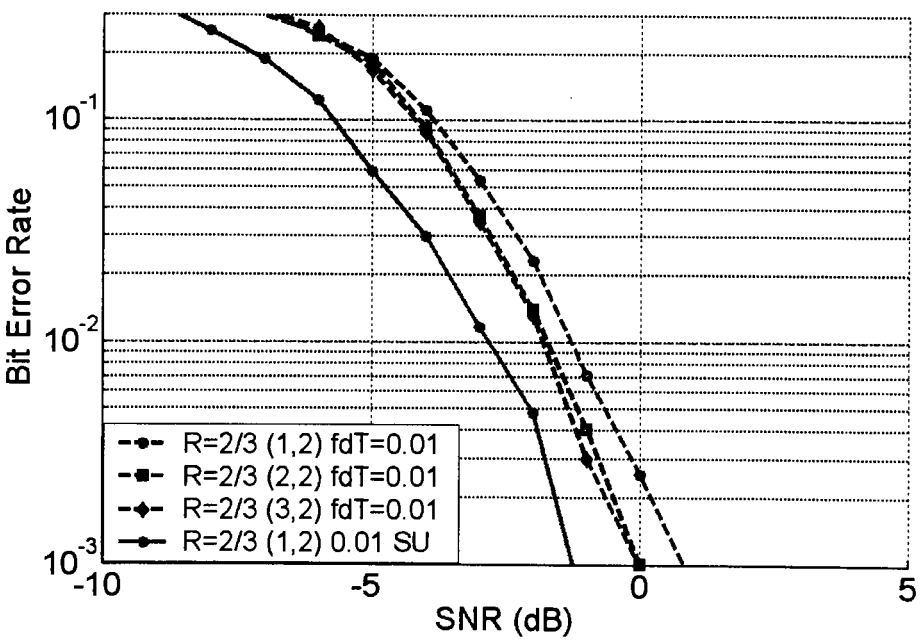


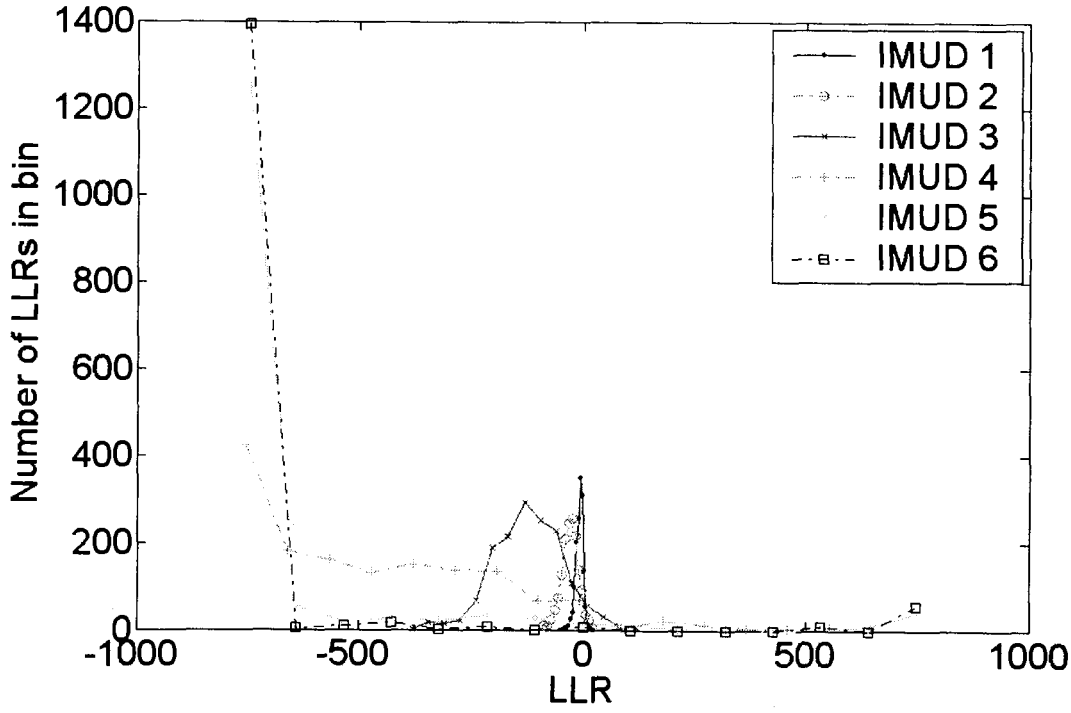

**Figure 40: (4,3,2) IMUD/SISO concatenated system with rate 2/3 code**



The results are encouraging. Like many iterative techniques, the biggest gain is achieved by the second iteration. However, unlike IMUD, the concatenated system continues to improve. The 4th and 5th iterations for the rate 1/3 code allow this observation; similar tests have not been run for the rate 2/3 code due to time constraints. This means that the APPs are still evolving after the first few iterations (Figures 41 and 42). Since these simulations only reveal the results after a few iterations are run, there is a possibility that a Turbo cliff region may appear after more iterations. However, longer block sizes may be necessary to uncover this phenomenon.

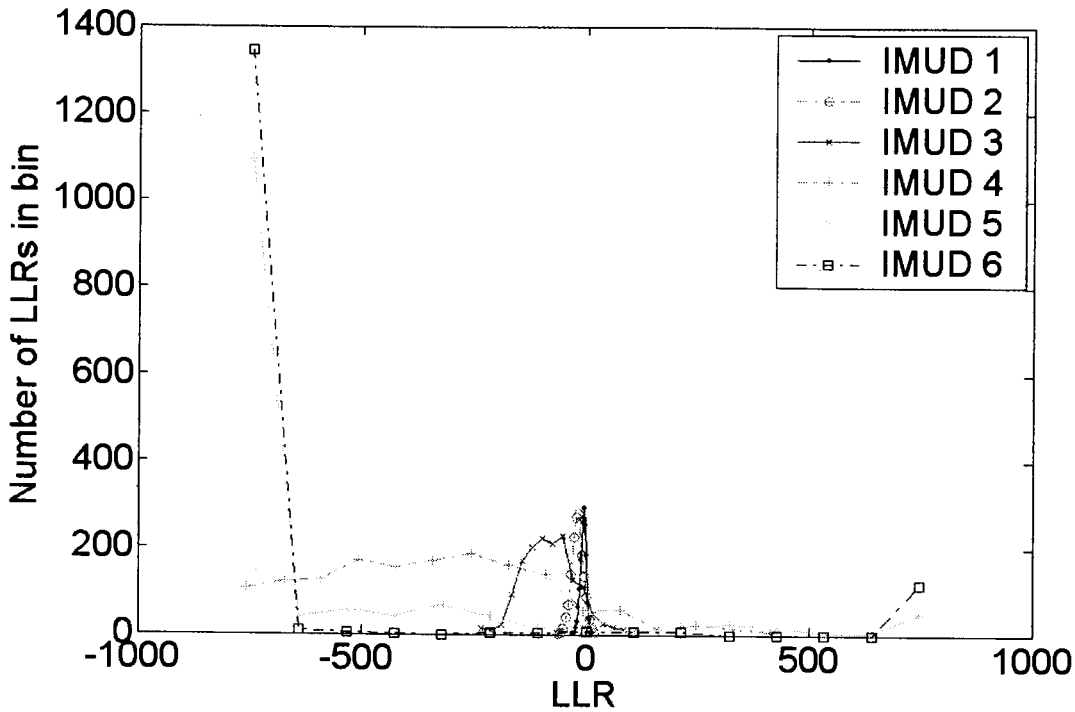
As further evidence of improvement with successive iterations, Figure 41 shows a histogram of the evolution of the LLRs at the output of the IMUD detector in a block of 100 coded symbols at an SNR of -7dB.

**Figure 41: Concatenated IMUD/SISO system, IMUD APP evolution at -7dB, rate 1/3 code**



100 symbols were transmitted with the rate 1/3 code, creating 300 LLR values total. A stream of -1 symbols were transmitted from the first user on the **(4,3,2)** system and is used to show how the APPs evolve over each iteration. All other users are transmitting a random stream. The ideal LLR for a +1 is $\infty$ and for a -1 is $-\infty$, so the decision boundary is at 0. To scale the plots, 800 was set as the maximum value for positive or negative value. Examining the first iteration (blue) of the IMUD/SISO system, it is seen that a few symbols are in error. However, with the next iteration (green), all the APPs are pulled closer to the ideal value of LLR. Again, for the third iteration (red), the LLRs are even further spread from the decision boundary. Figure 42 shows the same system at -10dB SNR. Some of the LLRs at this SNR are detected incorrectly, as seen in the positive LLRs at IMUD iterations 5 and 6. The BER plot in figure 39 is stated as evidence that the LLRs are not simply being pushed in the direction of the symbol that was initially detected.

**Figure 42: (4,3,2) concatenated IMUD/SISO system, IMUD APP evolution at -10dB, rate 1/3 code**



# 5 CONCLUSIONS

The topic of this thesis was to develop and assess a new multiuser detection (MUD) technique: Iterative Multiuser Detection (IMUD). IMUD was designed with influences from the optimum joint maximum likelihood (JML) MUD, the suboptimum linear V-BLAST MUD and group detection MUD. The novel developments in IMUD include the use of randomized groups with an iterative process and soft interference cancellation (IC) between groups and iterations. Through the testing of various parameters including ordering technique, probability feedback, and interference cancellation, the current performance of IMUD was maximized. This includes using error variance minimization for the ordering technique, strict *a posteriori* probability feedback between iterations, and interference cancellation between groups and iterations in IMUD.

The goal of IMUD was to perform as close as possible to the computationally prohibitive optimum MUD with complexity similar to existing suboptimum techniques. The performance of the IMUD simulation has been incredibly encouraging. In systems in which the number of single-antenna transmitting users equalled the number of receive antennas at the base station (Section 3.8.1), IMUD run with two iterations consistently outperformed both V-BLAST and the group detector with complexity in the same order of magnitude. For larger group sizes, this is of special interest since the optimum joint maximum likelihood technique quickly becomes prohibitive, with complexity many orders of magnitude above the suboptimal techniques. Even more impressive is IMUD's performance in an overloaded system (Section 3.8.2), where there are more transmitting users than receive antennas. Traditionally, the optimum JML technique has been the

only MUD to keep its order of diversity in an overloaded system. The suboptimal techniques develop error floors, making the techniques unusable in some cases. Although IMUD still exhibits this error floor, it is much lower than in other techniques. IMUD also preserves most of the diversity order (evident by the slope of the BER curve).

The use of IMUD can be extended beyond that of a stand-alone MUD, since the soft input/soft output nature of IMUD allows easy insertion in a serially concatenated system, similar to the equalizer in Turbo equalization. The IMUD/SISO system consists of users transmitting independent symbol streams that are encoded and interleaved, and a receiver consisting of an IMUD detector followed by a de-interleaver and decoder for each user or stream. This IMUD/SISO concatenated system was tested with 6 users and 4 receive antennas (Section 4.4). Rate 1/3 and rate 2/3 non-systematic codes were used. The resulting BER curves show that IMUD/SISO with 6 users and a rate 1/3 code can match an error rate of $10^{-3}$ of a single user system that has the same number of receive antennas, requiring an increase of only around 2dB in the SNR. Further tests are needed to understand how the error floor of the overloaded IMUD affects that of the IMUD/SISO system. However, for the simulated SNR range, the BER curve slope of the single-user and the 6 user IMUD/SISO system are nearly identical. For the rate 2/3 code, the difference between the 6 user case and the single user case is closer to 1dB.

## 5.1 Suggestions for Further Research

Since IMUD can be summarized simply as a soft input/soft output multiuser detector, there are many possible ways of utilizing it. For example, some immediate extensions of IMUD in a concatenated system include pairing it with space-time block codes, a macrodiversity system, or perhaps an OFDM system. All of these possibilities have potential for new innovations, especially the space-time system, where it uses the diversity of the channel in possibly complementary ways to IMUD.

Further investigation into IMUD itself will also be undertaken. Specifically, it is thought that the order of detecting users could be improved in two ways. First, there is the possibility of searching for an optimum ordering technique analytically, as well as via simulation. Second, the computational complexity should be examined, since from the plots in Section 3.7, it is evident that a major part of the complexity in IMUD is derived from the ordering technique.

For the convergence of IMUD, extrinsic transfer information (EXIT) charts may prove to be very useful [19]. EXIT charts are plots that show how mutual information is traded between components of iterative decoders. They can show how fast or even if the iterative system will converge to a final solution. In the IMUD/SISO case, the detector components for the EXIT chart would include IMUD and the SISO decoders. Another bonus in using EXIT charts is that they allow convergence of serial and parallel concatenated codes to be examined on a per iteration basis, without need to run Monte Carlo BER simulations. It is hoped that such charts can be compiled for different IMUD systems, allowing convergence to be analyzed between IMUD and various convolutional codes without the need for direct simulations.

# REFERENCE LIST


[1] J. G. Proakis, *Digital Communications*, 4th Ed. New York: McGraw Hill, 2001

[2] J. K. Cavers, *Mobile Channel Characteristics*. Boston: Kluwer, 2000.

[3] S. Verdu, "Minimum Probability of Error for Asynchronous Gaussian Multiple-Access Channels," *IEEE Trans. Inform. Theory*, vol. 32, pp 85-96, Jan. 1986.

[4] S. J. Grant, and J. K. Cavers, "Performance Enhancement Through Joint Detection of Cochannel Signals Using Diversity Arrays," *IEEE Trans. Commun.*, vol. 46, pp. 1038-1049, Aug. 1998.

[5] B. Vucetic, and J. Yuan, *Turbo Codes: Principles and Applications*, Boston: Kluwer, 2000.

[6] J. H. Winters, "Optimum Combining in Digital Mobile Radio with Cochannel Interference," *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 528-539, July 1984.

[7] S. Haykin, *Adaptive Filters*, 4th Ed. Upper Saddle Hill, NJ: Prentice-Hall, 2002.

[8] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An Architecture for Realizing Very High Data Rates Over the Rich-Scattering Wireless Channel," in *Proc. URSI International Symposium on Signals, Systems, and Electronics*, 1998, pp. 295-300.

[9] B. K. Ng, and E. S. Sousa, "On Bandwidth-Efficient Multiuser-Space-Time Signal Design and Detection," *IEEE J. Select. Areas Commun.*, vol. 20, pp. 320-329, Feb. 2002.

[10] S. W. Kim, "Log-Likelihood Ratio Based Detection Ordering for V-BLAST", in *IEEE Globecom 2003*, 2003, vol. 1, pg. 292-296.

[11] E. A. Fain, and M. K. Varanasi, "Diversity Order Gain for Narrow-Band Multiuser Communications with Pre-Combining Group Detection," *IEEE Trans. Commun.*, vol. 48, pp.533-536, April 2000.

[12] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284-287, March 1974.

[13] S. Benedetto, and G. Montorsi, "Iterative Decoding of Serially Concatenated Convolutional Codes," *IEEE Comm. Letters*, vol. 32, pp. 1186-1188, June 1996.

[14] M. Tuchler, R. Koetter, and A. C. Singer, "Turbo Equalization: Principles and New Results," *IEEE Trans. Commun.*, vol. 50, pp. 754-767, May 2002.

[15] H. V. Poor, "Iterative Multiuser Detection," *IEEE Sig. Proc. Mag.*, vol. 21, pp. 81-88, Jan. 2004.

[16] M. Sellathurai, and S. Haykin, "TURBO-BLAST for High-Speed Wireless Communications," in *Proc. WCNC 2000*, 2000, vol. 1, pp. 315-320.

[17] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A Soft-Input Soft-Output APP Module for Iterative Decoding of Concatenated Codes," *IEEE Commun. Lett.*, vol. 1, pp. 22-24, Jan. 1997.

[18] G. Colavolpe, G. Ferrari, and R. Raheli, "Extrinsic Information in Iterative Decoding: A Unified View," *IEEE Trans. Commun.*, vol. 49, pp. 2088-2094, Dec. 2001.

[19] S. ten Brink, "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," *IEEE Trans. Commun.*, vol. 49, pp. 1727-1737, Oct. 2001.