École polytechnique fédérale de Lausanne
Department of Communication Science

Master Degree in Communication Science

Multiple Access via Compute-and-Forward: Practical Code Design

Master Thesis of:
Erixhen Sula

Lab:
Laboratory for Information in Networked Systems (LINX)

Supervisors:
Prof. Michael Gastpar

Phd and Postdoc Supervisors:
Jingge Zhu, Sung Hoon Lim, Adriano Pastore

15 January 2016

Acknowledgments

I have been working under the supervision of Prof.Michael Gastpar, who is also the leader of the research group I am part of. I would like to thank him for the opportunity to be part of the research group. In this time frame I had the wonderful opportunity of being "initiated" to research, which radically changed the way I look at things: I found my natural *"thinking outside the box"* attitude — that was probably well-hidden under a thick layer of lack-of-opportunities.

It was a great pleasure to work with my phd supervisor Jingge Zhu for being among the prime suspects for getting me into all of this, and for his continuous support and guidance throughout the semester. As if this was not enough luck by itself, I also had the opportunity to interact with Sung Hoon Lim and Adriano Pastore, who influenced this work in many different ways.

Another important acknowledgment goes to all members of LINX group for the atmosphere and harmony, which is a team of outstanding researchers including here the secretary France Faille.

Erixhen Sula
Lausanne
January 2016

Abstract

In the *Multiple Access Channel* (MAC), we propose a novel *Sequential Decoding* (SD) strategy via *Compute-and-Forward* (CF) to achieve the boundary points on the capacity region. To investigate the practical potentials of this strategy we use *Low Density Parity Check* (LDPC) codes and address the problem of sequential decoding via compute-and-forward for binary linear codes, with reasonable complexity and good performance.

In a first part of the thesis, we recall the *Sum-Product Algorithm* (SPA) and its implementation. In the second part, we introduce the multiple access sequential decoder via CF for binary LDPC codes. In this part, this strategy is developed and analysed. For the two-user Gaussian MAC, it is shown that, points on the boundary of capacity region can be attained with reasonable complexity and we can achieve within 1.7 dB of the Shannon limit at a bit error rate of 10^{-4} using moderate block length of 1920. The strategy is extended to K-user Gaussian MAC. Our contribution regarding sequential decoding via compute-and-forward focuses on practical implementation with reasonable complexity, rarely implemented before. Simulation results are presented for binary and quaternary modulation, different user rates, and different channel gains.

The aforementioned strategy have been tested over one sample taken from Mackay's site — that are thoroughly documented in this thesis — and lead to interesting results.

Contents

List of Figures

List of Tables

List of Acronyms

SPA Sum-Product Algorithm ii

BP Belief Propagation 10

LDPC Low Density Parity Check ii

LLR Log Likelihood Ratio iv

SC Successive Cancellation 15

MAC Multiple Access Channel ii

AWGN Additive White Gaussian Noise 15

BPSK Binary Phase Shift Keying 22

MAP Maximum A Posteriori 18

ML Maximum Likelihood 18

PDF Probability Density Function 19

PAM Pulse Amplitude Modulation iv

SNR Signal-to-Noise Ratio vi

BER Bit Error Rate 22

CML Coded Modulation Library 23

CF Compute-and-Forward ii

SD Sequential Decoding ii

DM-MAC Discrete Memoryless Multiple Access Channel 16

Introduction 1

Linear codes, in particular LDPC codes are one of the most prominent examples of practical codes which we will use throughout the thesis. The LDPC codes have shown tremendous performance [1] for the point to point channel. However, practical codes have not been studied thoroughly for multiuser communication network, in particular the MAC.

In the MAC, all users share the same channel to transmit information to a destination and in particular the Gaussian MAC, which is a well-understood communication system. To achieve the boundary points on capacity region, the receiver can either use joint decoding, or a single-user decoder combined with successive cancellation decoding. Moreover, to achieve the entire capacity region the above technique is combined with Rate-Splitting Multiple Access in [2], but at the price that messages have to be split to create virtual users. Selecting the proper decoder is a trade-off between performance and complexity. The purpose of our project is to design an efficient and low complexity decoder, and for this reason we will focus on sequential decoding via CF technique.

In this thesis we provide and analyse a new strategy for the Gaussian MAC using linear codes based on CF technique [3]. For the 2-user Gaussian MAC case, the receiver first decodes the modulo sum of the two codewords, and then decodes either one of the codewords using the sum as side information. Each step of sequential decoding via CF technique has a complexity in the order of a point to point decoder.

Hence this new approach allows us to achieve rate pairs in the capacity region using sequential decoding via CF. The proposed decoding strategy is then extended to the general case of K-user Gaussian MAC. A complete characterisation of the achievable rates region is given in [4] for the general case.

The thesis is organised as follows. Chapter 2 introduces the linear codes, in particular LDPC used in our decoding strategy. Also SPA is explained together with the approximations made to lower complexity. Chapter 3 gives

the problem statement, a complete analysis of our decoding strategy for the 2-user Gaussian MAC, and an extension to K-user Gaussian MAC. Moreover, in this chapter other extensions are considered as well. In addition, all the simulation results are included in this chapter after every theoretical treatment.

Random codes are often used as a proof technique in the information-theoretic literature. However, random codes are impractical because they lack structure, in the sense that they do not admit a concise representation other than an explicit list of codewords. Linear codes are defined over alphabets Σ which are finite fields. Throughout, we will denote by $\mathbb{F}_q$ the finite field with q elements, where q is a prime power. For now, we can think of q as a prime, in which case $\mathbb{F}_q$ is just $\{0, 1, \ldots, q-1\}$ with addition and multiplication defined modulo q.

Definition 2.0.1 (Linear Codes) *If Σ is a field and $\mathcal{C} \subseteq \Sigma^n$ is a subspace of Σ^n then $\mathcal{C}$ is said to be a linear code.*

As $\mathcal{C}$ is a subspace, there exists a basis $c_1, c_2, \ldots, c_k$ where k is the dimension of the subspace. Any codeword can be expressed as the linear combination of these basis vectors. We can write these vectors in matrix form as the rows of a $k \times n$ matrix. Such a matrix is called a generator matrix.

Definition 2.0.2 (Generator Matrix) *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a linear code of dimension k. A matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ is said to be a generator matrix for $\mathcal{C}$ if its k rows span $\mathcal{C}$. The generator matrix $\mathbf{G}$ provides a way to encode a message $x \in \mathbb{F}_q^k$ (thought of as a row vector) as the codeword $x\mathbf{G} \in \mathcal{C} \subseteq \mathbb{F}_q^n$. Thus a linear code has an encoding map $Enc : \mathbb{F}_q^k \mapsto \mathbb{F}_q^n$ which is a linear transformation $x \mapsto x\mathbf{G}$.*

Many coding textbooks and papers define the "transposed" version, where the columns of the $n \times k$ generator matrix span the code. Note that a linear code admits many different generator matrices, corresponding to the different choices of basis for the code as a vector space. However, this does not change the structure of the code at all.

Definition 2.0.3 (Parity Check Matrix) *Let $\mathcal{C}$ be an (n,k)-code over $\mathbb{F}_q^n$. There exists an $(n-k)\times n$-matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ which is of rank $n-k$ and satisfies*

$$\mathcal{C} = \left\{ w \in \mathbb{F}_q^n \quad | \quad w\mathbf{H}^\top = 0 \right\},$$

where $\mathbf{H}^\top$ denotes the transpose of the matrix $\mathbf{H}$. Any such matrix is called check matrix of $\mathcal{C}$. [5]

The *rate* of the codebook $\mathcal{C}_k$ is defined to be:

$$R_k = \frac{1}{n} \log |\mathcal{C}| \cdot \log q \quad \text{for } k = 1, 2$$

where n represent the length of the message and q is the finite field size.

2.1 Nested Codes

Nested codes are always associated with multiple access scenario, considering the definition of nested codes. This section establishes the basic definition of nested codes. We start with the binary case and nested parity-check codes. More formally stated a nested code is a pair of linear or lattice codes $(\mathcal{C}_1, \mathcal{C}_2)$ satisfying

$$\mathcal{C}_2 \subseteq \mathcal{C}_1 \tag{2.1}$$

We call $\mathcal{C}_1$ the "fine code" and $\mathcal{C}_2$ the "coarse code". If a pair $\{(n,k_1),(n,k_2)\}$ of codes where $k_1 > k_2$, satisfies condition (2.1), then the corresponding parity check matrices $\mathbf{H}_1$ and $\mathbf{H}_2$ are interrelated as

$$\mathbf{H_2} = \begin{bmatrix} \mathbf{H_1} \\ \mathbf{\Delta H} \end{bmatrix} \tag{2.2}$$

where $\mathbf{H}_1$ is $(n-k_1)\times n$ matrix, $\mathbf{H}_2$ is $(n-k_2)\times n$ and $\mathbf{\Delta H}$ is $(k_1-k_2)\times n$ matrix. This implies that syndromes $s_1 = x\mathbf{H}_1^\top$ and $s_2 = x\mathbf{H}_2^\top$ associated with the n-length vector x are related as in [6] $s_2 = [s_1, \Delta s]$ where the length of Δs is $k_1 - k_2$ bits. If $s_2 = \mathbf{0}$ it implies that $s_1 = \mathbf{0}$, in other words every codeword of codebook $\mathcal{C}_2$ is also a codeword of codebook $\mathcal{C}_1$. In particular, if $x \in \mathcal{C}_1$, then $s_2 = [\mathbf{0}, \Delta s]$. Therefore, we can partition $\mathcal{C}_1$ into $2^{k_1-k_2}$ cosets of $\mathcal{C}_2$ by setting $s_1 = \mathbf{0}$, and varying Δs.

$$\mathcal{C}_1 = \bigcup_{\Delta s \in \{0,1\}^{k_1-k_2}} \mathcal{C}_{2,s_2}$$

Of fundamental importance is the question: can we require both components of a nested code, the fine code and the coarse code, to be both "good" performing?

2.2 Low Density Parity Check Codes

The name comes from the characteristic of their parity-check matrix which contain only a few 1's in comparison to the amount of 0's which in other words

is a sparse matrix. Their main advantage is that they provide a performance which is actually very close to the Shannon capacity for a lot of different channels and low complexity algorithms for decoding. Furthermore they are suited for implementations that make heavy use of parallelism. They were first introduced by Gallager in his PhD thesis in 1960.

2.2.1 Matrix Representation

Let us consider an example for a low density parity check matrix first. The matrix defined in equation (2.3) is a parity check matrix with dimension $n \times (n-k)$ specifically 8×4 code. We denote by w_r and w_c as the number of 1's in each row and columns, respectively. For a matrix to be called low-density the two conditions $w_c \ll n$ and $w_r \ll m$ must be satisfied. Nonetheless, for pedagogical reasons, we will make use of the following simple example.

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{2.3}$$

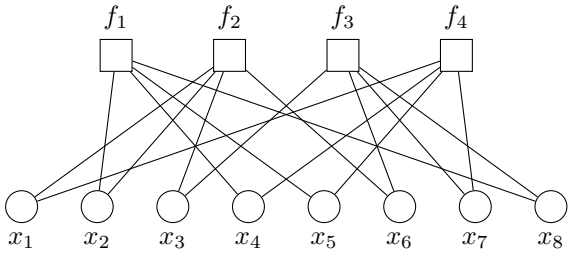

FIGURE 2.1: Tanner graph corresponding to the parity check matrix.

2.2.2 Graphical Representation

Tanner introduced an effective graphical representation for LDPC codes. Not only do these graphs provide a complete representation of the code, they also help to describe the decoding algorithm. Tanner graphs are bipartite graphs. That means that the nodes of the graph are separated into two distinctive sets, and edges do only connect nodes of two different types. The two types of nodes in a Tanner graph are called variable nodes (f-nodes) and check nodes (x-nodes). Figure (2.1) is an example of such a Tanner graph and represents graphically the code expressed in matrix form (2.3). The creation of such a graph is rather straightforward. It consists of $n-k$ check nodes (the number of parity bits) and n variable nodes (the number of bits in a codeword). Check node f_i is connected to variable node x_j if the element h_{ij} of $\mathbf{H}$ is a 1.

2.3 Sum Product Algorithm

We will introduce factor graphs to describe a generic message-passing algorithm, called SPA. The SPA is a basic algorithm and a wide variety of variants have been developed in signal processing, digital communications, and other communities. Factor graphs are a straightforward generalisation of "Tanner graphs". Tanner in [7] introduced bipartite graphs to describe families of codes which are generalisations of Gallager's LDPC codes in [8].

The decoding complexity can be reduced dramatically when the factor graph has some special structure. One particular case is that of tree factor graphs. On trees, marginals can be computed in a number of operations which grows linearly with the depth of the tree. This can be done through a procedure that recursively sums over all variables starting from the leaves and progressing towards the root of the tree. It is straightforward to prove that belief propagation exactly computes marginals on tree factor graphs.

We begin with an example that illustrates the operation of the sum product algorithm in a simple factor graph in [9].

2.3.1 Factor Graphs

Let $x_1, x_2, \ldots, x_n$ be the set of variable nodes where the i-th element x_i takes values from the domain A_i. In the case of a binary finite field the domain is $A_i = \{0, 1\}$. Now consider the global function of these variables $g(x_1, x_2, \ldots, x_n)$ with domain $S = A_1 \times A_2 \times \cdots \times A_n$ where the range of this function is $\mathbb{R}$. Our global function $g(x_1, x_2, \ldots, x_n)$ has n marginal functions $g_i(x_i)$ defined as:

$$g_i(x_i) = \sum_{\sim x_i} g(x_1, x_2, \ldots, x_n) \tag{2.4}$$

where $\sim x_i$ denotes the summation over the set $\{x_1, x_2, \ldots, x_n\} \setminus x_i$. Suppose that $g(x_1, x_2, \ldots, x_n)$ factors into a product of several *local functions*, each having some subset of $\{x_1, x_2, \ldots, x_n\}$ as arguments:

$$g(x_1, x_2, \ldots, x_n) = \prod_{j \in J} f_j(X_j) \tag{2.5}$$

where J is a discrete index set and X_j is a subset of $\{x_1, x_2, \ldots, x_n\}$.

Definition 2.3.1 (Factor Graph) *A factor graph is a bipartite graph that expresses the structure of the equation* (2.5)*. A factor graph has a variable node for each variable x_i, a factor node for each local function f_j, and an edge connecting variable node x_i to factor node f_j if and only if x_i is an argument of f_j.*

Example 2.3.1 (A Simple Factor Graph) *Let $g(x_1, x_2, x_3, x_4, x_5)$ be a function of 5 variables and suppose that g can be expressed as a product of 5 factors so that $J = \{A, B, C, D, E\}$, $X_A = \{x_1\}$, $X_B = \{x_2\}$, $X_C = \{x_1, x_2, x_3\}$,*

$X_D = \{x_3, x_4\}$ *and* $X_E = \{x_4, x_5\}$. *The graph that corresponds to equation* (2.6) *is shown in the figure*

$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5) \quad (2.6)$$

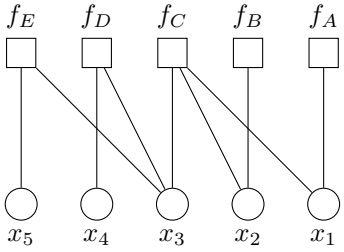

FIGURE 2.2: Factor or Tanner graph for the global function g of equation (2.6)

2.3.2 Update Rule

Let $\mu_{x \to f}$ denote the message sent from variable node x to check node f in the operation of the sum product algorithm, let $\mu_{f \to x}$ denote the message sent from check node f to variable node x. Also, let $n(x)$ denote the set of neighbours of a given node x in a factor graph. Then, the message computations performed by the sum product algorithm may be expressed as follows:

variable to check node:

$$\mu_{x \to f}(x) = \prod_{h \in n(x) \setminus \{f\}} \mu_{h \to x}(x) \quad (2.7)$$

check to variable node:

$$\mu_{f \to x}(x) = \sum_{\sim \{x\}} \left(f(X) \prod_{y \in n(f) \setminus \{x\}} \mu_{y \to f}(y) \right) \quad (2.8)$$

marginal update:

$$g(x) = \prod_{h \in n(x)} \mu_{h \to x}(x) \quad (2.9)$$

where X is the set of arguments of function f. The proof of these update rules is a direct consequence of transformation of expression tree and specifically looking only to the neighbours of a variable node or a check node. More details can be found in [9]. To better understand how the update rules operate on a factor or Tanner graph we will give an example with the complete update steps. But, before that let me give some further information about application of update rules. When considering the so called "leaf" of a tree the update rules simplify as follows:

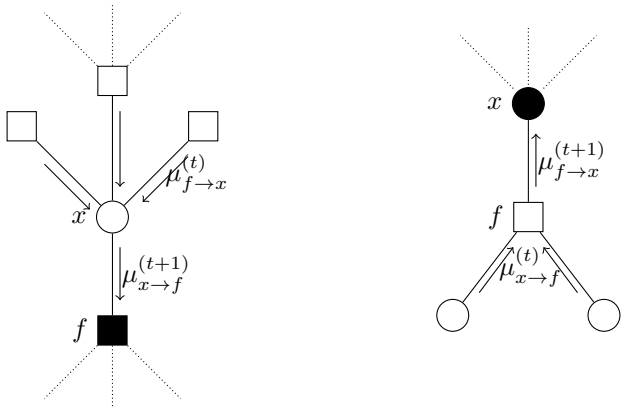

Figure 2.3: Left: portion of the factor graph involved in the computation of $\mu_{x\to f}^{(t+1)}$ where t denoted the step number. This message is a function of $\mu_{x\to f}^{(t)}$. Right: portion of the factor graph involved in the computation of $\mu_{f\to x}^{(t+1)}$. This message is a function of $\mu_{f\to x}^{(t)}$.

variable node x is a leaf:

$$\mu_{x\to f}(x) = 1 \tag{2.10}$$

check node f is a leaf:

$$\mu_{f\to x}(x) = f(x) \tag{2.11}$$

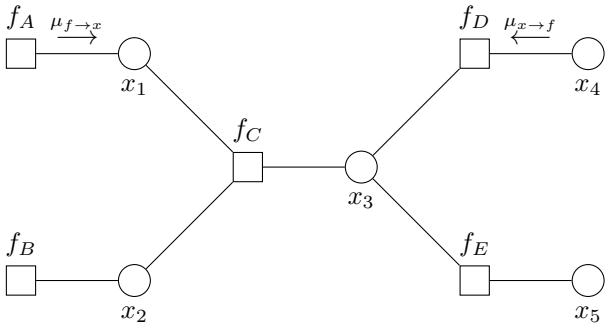

Figure 2.4: Factor graph for the global function g of equation (2.6)

Example 2.3.2 (Update Rule Application) *Let $g(x_1, x_2, x_3, x_4, x_5)$ be a function of 5 variables and suppose that g can be expressed as a product of 5 factors as described in the figure 2.4. Message passing algorithm has a flow of update rules as shown below:*

1^{st} Step:

$$\mu_{f_A \to x_1}(x_1) = \sum_{\sim\{x_1\}} f_A(x_1) = f_A(x_1)$$
$$\mu_{f_B \to x_2}(x_2) = \sum_{\sim\{x_2\}} f_B(x_2) = f_B(x_2)$$
$$\mu_{x_4 \to f_D}(x_4) = 1$$
$$\mu_{x_5 \to f_E}(x_5) = 1$$

2^{nd} Step:

$$\mu_{f_D \to x_3}(x_3) = \sum_{\sim\{x_3\}} \mu_{x_4 \to f_D}(x_4) f_D(x_3, x_4)$$
$$\mu_{f_E \to x_3}(x_3) = \sum_{\sim\{x_3\}} \mu_{x_5 \to f_E}(x_5) f_E(x_3, x_5)$$
$$\mu_{x_1 \to f_C}(x_1) = \mu_{f_A \to x_1}(x_1)$$
$$\mu_{x_2 \to f_C}(x_2) = \mu_{f_B \to x_2}(x_2)$$

3^{rd} Step:

$$\mu_{f_C \to x_3}(x_3) = \sum_{\sim\{x_3\}} \mu_{x_1 \to f_C}(x_1) \mu_{x_2 \to f_C}(x_2) f_C(x_1, x_2, x_3)$$
$$\mu_{x_3 \to f_C}(x_3) = \mu_{f_D \to x_3}(x_3) \mu_{f_E \to x_3}(x_3)$$

4^{th} Step:

$$\mu_{f_C \to x_1}(x_1) = \sum_{\sim\{x_1\}} \mu_{x_3 \to f_C}(x_3) \mu_{x_2 \to f_C}(x_2) f_C(x_1, x_2, x_3)$$
$$\mu_{f_C \to x_2}(x_2) = \sum_{\sim\{x_2\}} \mu_{x_3 \to f_C}(x_3) \mu_{x_1 \to f_C}(x_1) f_C(x_1, x_2, x_3)$$
$$\mu_{x_3 \to f_D}(x_3) = \mu_{f_C \to x_3}(x_3) \mu_{f_E \to x_3}(x_3)$$
$$\mu_{x_3 \to f_E}(x_3) = \mu_{f_C \to x_3}(x_3) \mu_{f_D \to x_3}(x_3)$$

5^{th} Step:

$$\mu_{f_D \to x_4}(x_4) = \sum_{\sim\{x_4\}} \mu_{x_3 \to f_D}(x_3) f_D(x_3, x_4)$$
$$\mu_{f_E \to x_5}(x_5) = \sum_{\sim\{x_5\}} \mu_{x_3 \to f_E}(x_3) f_E(x_3, x_5)$$
$$\mu_{x_1 \to f_A}(x_1) = \mu_{f_C \to x_1}(x_1)$$
$$\mu_{x_2 \to f_B}(x_2) = \mu_{f_C \to x_2}(x_2)$$

Now the most important step is to compute the marginals $g_i(x_i)$ which we want to express as a product of update messages which in log domain will reduce to a sum so that complexity will be reduced significantly.

Marginalization Step:

$$
\begin{aligned}
g_1(x_1) &= \mu_{f_A \to x_1}(x_1)\mu_{f_C \to x_1}(x_1) \\
g_2(x_2) &= \mu_{f_B \to x_2}(x_2)\mu_{f_C \to x_2}(x_2) \\
g_3(x_3) &= \mu_{f_C \to x_3}(x_3)\mu_{f_D \to x_3}(x_3)\mu_{f_E \to x_3}(x_3) \\
g_4(x_4) &= \mu_{f_D \to x_4}(x_4) \\
g_5(x_5) &= \mu_{f_E \to x_5}(x_5)
\end{aligned}
$$

2.3.3 Particular Factor Graphs

With this example we now have a clear idea on how the message passing in *Belief Propagation* (BP) is done iteratively back and forth between the variable nodes and check nodes of the factor graph. However the example 2.3.2 describes only a single update step and in order to compute the exact marginals in case when the factor graph has a tree structure we need t^* step where t^* represents the maximum distance between 2 variable nodes.

Theorem 2.3.1 (BP is exact) *Consider a tree factor graph with depth t^* (which means that t^* is the maximum distance between any two variable nodes). Then*

1. *Irrespective of the initial condition, the BP update* (2.7), (2.8) *converges after at most t^* iterations. In other words, for any $t > t^*$ $\mu^{(t)}_{f \to x} = \mu^*_{f \to x}$ and $\mu^{(t)}_{x \to f} = \mu^*_{x \to f}$*
2. *For any variable node x and $t > t^*$, $\mu^{(t)}_x = g(x)$*

Proof: Let us sketch an informal proof, leaving details to the reader. This claim is proved by induction on the tree depth t^*. The induction step is easy as well. Assuming the claim to be true for $t^* \leq \tau$, one has to show that it holds when $t^* = \tau + 1$.

2.4 Implementation of SPA

The main concern of this section is how to implement SPA algorithm such that the running time or complexity remains as low as possible. In order to achieve this one cannot use directly the update rule (2.7) and (2.8) because the running time can be arbitrarily large. As we are looking for a more practical solution, some reasonable assumptions and approximations should take place. Let us consider the case when there is no log-likelihood ratio and introduce some notation which are used widely in the literature related to SPA algorithm.

- q_{ij} is a message sent by the variable node x_i to the check node f_j. Every message contains always the pair $q_{ij}(0)$ and $q_{ij}(1)$ which stands for the amount of belief that y_i is 0 or 1. Throughout this work we will operate only with binary linear codes, which means that messages can be in the set $\{0, 1\}$.
- r_{ji} is a message sent by the check node f_j to the variable node x_i. Again there is a $r_{ji}(0)$ and $r_{ji}(1)$ that indicates the amount of belief in that y_i is 0 or 1.
- C_i is the set of all "neighbour" check nodes connected to variable node x_i.
- V_j is the set of all "neighbour" variable nodes connected to check node f_i.
- k_{ij} is only a constant to normalize the update rules.
- $$m_{ij} = \log \frac{q_{ij}(0)}{q_{ij}(1)} \qquad \text{whereas} \qquad n_{ji} = \log \frac{r_{ji}(0)}{r_{ji}(1)}$$

Then the update rule (2.7) will take this form:

$$q_{ij}^{(t+1)}(0) = k_{ij} \prod_{j' \in C_i \setminus j} r_{j'i}^{(t)}(0) \tag{2.12}$$

$$q_{ij}^{(t+1)}(1) = k_{ij} \prod_{j' \in C_i \setminus j} r_{j'i}^{(t)}(1) \tag{2.13}$$

By taking the log-ratio of equation (2.12) and (2.13) we obtain:

$$m_{ij}^{(t+1)} = \log \frac{q_{ij}^{(t+1)}(0)}{q_{ij}^{(t+1)}(1)} = \log \prod_{j' \in C_i \setminus j} \frac{r_{j'i}^{(t)}(0)}{r_{j'i}^{(t)}(1)} = \sum_{j' \in C_i \setminus j} n_{j'i}^{(t)} \tag{2.14}$$

This step is quite easy to implement due to the low complexity of the algebraic summation operation. Thus, no approximation is required in this case. Now let us focus on the second update rule (2.8). This step contains an additional function $f(x)$ which represents the check node function defined in this way:

$$f(x) = \mathbb{1}_{\{x\mathbf{H}^\top = \mathbf{0}\}}$$

The explicit expression of $f(x)$ will be proven at a later stage. From now on let us assume $f(x)$ is defined as above where $\mathbf{H}$ is an $(n-k)\times n$ matrix and let $\mathbf{H} = [f_1, f_2, \ldots, f_{n-k}]^\top$ thus $f(x) = \prod_{j=1}^{n-k} \mathbb{1}_{\{xf_j^\top = \mathbf{0}\}}$. Then the update rule (2.8) will take this form:

$$r_{ji}^{(t+1)} = \sum_{x_{i'}} \left(\mathbb{1}_{\{xf_j^\top = \mathbf{0}\}} \prod_{i' \in V_j \setminus i} q_{i'j}^{(t)} \right) \tag{2.15}$$

Proposition 2.4.1 *The equation* (2.15) *has the following explicit form after some manipulation:*

$$r_{ji}^{(t+1)}(0) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in V_j \setminus i} \left(1 - 2q_{i'j}^{(t)}(1)\right) \tag{2.16}$$

$$r_{ij}^{(t+1)}(1) = \frac{1}{2} - \frac{1}{2} \prod_{i' \in V_j \setminus i} \left(1 - 2q_{i'j}^{(t)}(1)\right) \tag{2.17}$$

Proof: This claim is proven by induction on length of f and first of all let us consider the case when vector f has all entries equal to 1 and length 3:

$$\begin{aligned} r_{j3}^{(t+1)}(0) &= \sum_{x_1,x_2} \left(\mathbb{1}_{\{x_1 \oplus x_2 \oplus 0 = 0\}} \prod_{i' \in \{1,2\}} q_{i'j}^{(t)} \right) \\ &= q_{1j}^{(t)}(0) q_{2j}^{(t)}(0) + q_{1j}^{(t)}(1) q_{2j}^{(t)}(1) \\ &= q_{1j}^{(t)}(1) q_{2j}^{(t)}(1) + \left(1 - q_{1j}^{(t)}(1)\right)\left(1 - q_{2j}^{(t)}(1)\right) \\ &= \frac{1}{2} + \frac{1}{2}\left(1 - 2q_{1j}^{(t)}(1)\right)\left(1 - 2q_{2j}^{(t)}(1)\right) \\ &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in \{1,2\}} (1 - 2q_{i'j}^{(t)}(1)) \end{aligned}$$

Now to give the complete proof we assume that equation (2.15) is true when the length of f is $n-1$ and then prove it for length n. We omit the proof, which is similar to the one before. Afterward by analogy we derive the final form for any vector f.

By taking the log-ratio of equation (2.16) and (2.17) we obtain:

$$\begin{aligned} n_{ji}^{(t+1)} &= \log \frac{r_{ji}^{(t+1)}(0)}{r_{ji}^{(t+1)}(1)} \\ &= \log \frac{\frac{1}{2} + \frac{1}{2}\prod_{i' \in V_j \setminus i} \left(1 - 2q_{i'j}^{(t)}(1)\right)}{\frac{1}{2} - \frac{1}{2}\prod_{i' \in V_j \setminus i} \left(1 - 2q_{i'j}^{(t)}(1)\right)} \\ &\overset{(a)}{=} \log \frac{1 + \prod_{i' \in V_j \setminus i} \tanh\left(\frac{m_{i'j}^{(t)}}{2}\right)}{1 - \prod_{i' \in V_j \setminus i} \tanh\left(\frac{m_{i'j}^{(t)}}{2}\right)} \\ &= 2\tanh^{-1}\left(\prod_{i' \in V_j \setminus i} \tanh\left(\frac{m_{i'j}^{(t)}}{2}\right)\right) \end{aligned}$$

where (a) follows by the argument in equation (2.14) where

$$q_{i'j}(1) = \frac{1}{1 + e^{m_{i'j}}} \qquad \Rightarrow \qquad 1 - 2q_{i'j}(1) = \frac{e^{m_{i'j}} - 1}{e^{m_{i'j}} + 1} = \tanh\left(\frac{m_{i'j}^{(t)}}{2}\right)$$

The problem with these expressions is that even operating in log-domain we cannot get rid of the product. Also it involves a hyperbolic tangent and its inverse. How do we proceed? We do some approximations in order to keep the complexity as low as possible as in [10]. First, split n_{ji} into its sign and magnitude:

$$\begin{aligned} m_{ij} &= \alpha_{ij}\beta_{ij} \\ \alpha_{ij} &= \text{sign}[m_{ij}] \\ \beta_{ij} &= |m_{ij}| \end{aligned}$$

Then we have (for the moment we omit the step index for notation brevity):

$$\begin{aligned} n_{ji} &= \prod_{i'} \alpha_{i'j} \cdot 2\tanh^{-1}\left(\prod_{i' \in V_j \setminus i} \tanh\left(\frac{\beta_{i'j}}{2}\right)\right) \\ &= \prod_{i'} \alpha_{i'j} \cdot 2\tanh^{-1} \exp\log\left(\prod_{i' \in V_j \setminus i} \tanh\left(\frac{\beta_{i'j}}{2}\right)\right) \\ &= \prod_{i'} \alpha_{i'j} \cdot 2\tanh^{-1} \exp\sum_{i' \in V_j \setminus i} \log\left(\tanh\left(\frac{\beta_{i'j}}{2}\right)\right) \\ &= \prod_{i'} \alpha_{i'j} \cdot \phi_0^{-1}\left(\sum_{i' \in V_j \setminus i} \phi_0(\beta_{i'j})\right) \end{aligned}$$

where ϕ_0 is defined as:

$$\phi_0(x) = -\log\tanh\left(\frac{x}{2}\right) = \log\left(\frac{e^x+1}{e^x-1}\right)$$

and use the fact that $\phi_0^{-1}(x) = \phi_0(x)$ when $x > 0$.

2.4.1 Approximation of function ϕ_0

This function is well-behaved and has the following look-up table:

x	$\phi_0^*(x)$
(0,9.08e-5]	10
(9.08e-5,5]	$\log\left(\frac{e^x+1}{e^x-1}\right)$
(5,6]	9.2168e-3
(6,7]	3.3906e-3
(7,8]	1.2473e-3
(8,9]	4.5887e-4
(9,10]	1.6881e-4
>10	0

Table 2.1: Approximation $\phi_0^*(x)$ of the auxiliary function $\phi_0(x)$ for different values of x when $x > 0$

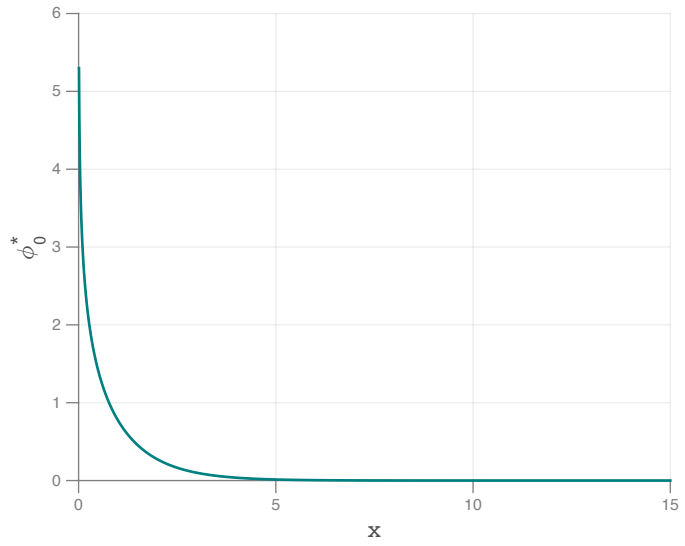

Figure 2.5: Table look-up approximation of the auxiliary function $\phi_0(x)$

The *Successive Cancellation* (SC) is a multiuser decoding technique which can be used to achieve the boundary of the capacity region of a MAC as in [11]. SC decoding is based on subtracting (cancelling) the already decoded codeword from the received codeword before the decoding of the next codeword. For example, in the case of 2-user Gaussian MAC the first step is to decode one of the codewords in which the other codeword is treated as noise, while in the second step the decoded codeword is subtracted. The SC is very powerful and looking from complexity perspective it is in the same order as point to point decoding (when channel is *Additive White Gaussian Noise* (AWGN)). On the other hand, sequential decoding via CF technique is also used to achieve the boundary of the capacity region of linear codes.

An immediate question that raises here is: What is the difference between SC and sequential decoding via CF? As mentioned before both technique are used for achieving points on the boundary of capacity region however, SD via CF is used for the points which cannot be decoded by SC. Moreover, SD via CF is used to recover a linear functions of codewords without recovering the individual codewords. In general, it decodes modulo sum of the codewords without treating any codeword as noise and as a second step decodes one of the codewords individually using the sum of codewords as side information. Our main goal now is to do SD via CF using efficient decoding algorithm.

3.1 Problem Statement

Consider the multiple access communication system model as shown in figure 3.1. Each sender communicates an *independent* message reliably to a single receiver. Sender $j = \{1, 2\}$ encodes its message W_j into a codeword X_j^n and it transmits it over the channel. After receiving the sequence Y^n, the decoder finds estimates $\hat{W}_j$ of each message. Since the senders transmit over a noisy

channel, a tradeoff arises between the rates of reliable communication for the two messages.

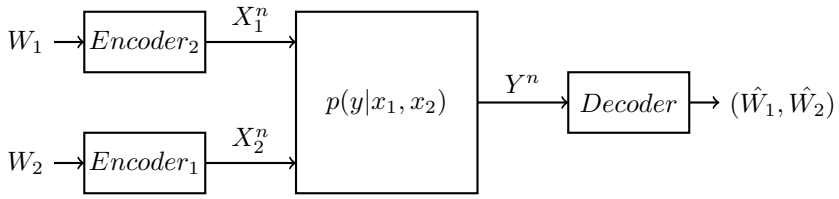

Figure 3.1: Multiple access communication system with independent messages.

We first consider a 2-sender *Discrete Memoryless Multiple Access Channel* (DM-MAC) model $(\mathcal{X}_1 \times \mathcal{X}_2, p(y|x_1, x_2), \mathcal{Y})$ that consists of conditional pmfs $p(y|x_1, x_2)$ on $\mathcal{Y}$ (one for each input symbol pair (x_1, x_2)).

A $(2^{nR_1}, 2^{nR_2}, n)$ code for the DM-MAC consists of:

- two message sets $[1 : 2^{nR_1}]$ and $[1 : 2^{nR_1}]$,
- two encoders, where encoder 1 assigns a codeword $x_1^n(w_1)$ to each message $w_1 \in [1 : 2^{nR_1}]$ and encoder 2 assigns a codeword $x_2^n(w_2)$ to each message $w_2 \in [1 : 2^{nR_2}]$, and
- a decoder that assigns an estimate $(\hat{w}_1, \hat{w}_2) \in [1 : 2^{nR_1}] \times [1 : 2^{nR_2}]$ or an error message to each received sequence y_n.

We assume that the message pair (M_1, M_2) is uniformly distributed over $[1 : 2^{nR_1}] \times [1 : 2^{nR_2}]$. Consequently, $x_1^n(W_1)$ and $x_2^n(W_2)$ are independent. The average probability of error is defined as

$$P_e^{(n)} = P\left\{(\hat{W}_1, \hat{W}_2) \neq (W_1, W_2)\right\}$$

A rate pair (R_1, R_2) is said to be *achievable* for the DM-MAC if there exist a sequence of $(2^{nR_1}, 2^{nR_2}, n)$ codes such that $\lim_{n\to\infty} P_e^{(n)} = 0$. The *capacity region* $\mathscr{C}$ of the DM-MAC is the closure of the set of achievable rate pairs (R_1, R_2). Sometimes we are interested in the sum-capacity $\mathcal{C}_{sum}$ of the DM-MAC defined as $\mathcal{C}_{sum} = \max\{R_1 + R_2 : (R_1, R_2) \in \mathscr{C}\}$.

3.2 Linear Codes Capacity

An important question here would be: What is the achievable rate pair (R_1, R_2) that linear codes can reach? In order to answer the question we need to analyse the *simultaneous decoder* for linear codes. This step is not as easy as it looks in the sense that the analysis of the probability of error for linear codes is closely related to the analysis of the random codes. However, due to the dependency issues (in linear codes) the packing lemma in [12] for i.i.d random codes cannot be applied directly in our case. To solve this issue a lemma which is adapted for linear codes is found in [13]. Now I will state a theorem which determines

the achievable rates for linear codes. Since we need only the theorem itself, I will only cite the proof.

Theorem 3.2.1 *A rate pair (R_1, R_2) is achievable by linear random codes over the discrete memoryless MAC if it is included in*

$$\bigcup_{a,b\in\mathbb{F}_q^2} \mathcal{R}_1(a,b) \tag{3.1}$$

for some prime q, $p(u_1,x_1)p(u_2,x_2)$ where $a=[a_1,a_2]$, $b=[b_1,b_2]$, the union is over all $a,b\in\mathbb{F}_q^2$ such that $a\neq 0$ and $b\neq 0$ are linearly independent,

$$\begin{aligned}
&\mathcal{R}_1(a,b) = \{(R_1,R_2):\\
&R_1 < I(U_1;Y|U_2),\\
&R_2 < I(U_2;Y|U_1),\\
&R_1+R_2 < I(U_1,U_2;Y),\\
&\min\{R_1+I(W_1;U_2), R_2+I(W_1;U_1)\} < I(W_1;Y)\\
&\min\{R_1+I(W_2;U_2), R_2+I(W_2;U_1)\} < I(W_2;Y,W_1)
\end{aligned} \tag{3.2}$$

where

$$\begin{bmatrix} W_1 \\ W_2 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{3.3}$$

Proof 3.2.1 *The complete proof can be found in [13]*

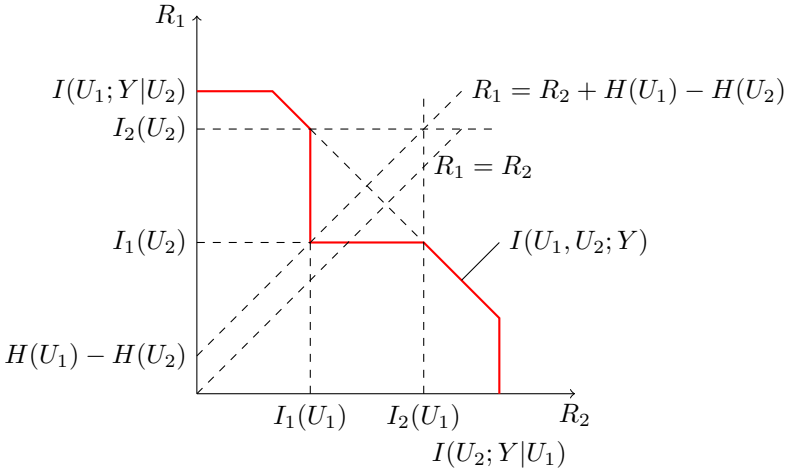

FIGURE 3.2: An example of the rate region $\mathcal{R}_1(a,b)$ for some fixed q, $p(u_1,x_1)p(u_2,x_2)$ and a,b where $H(U_1) > H(U_2)$

Where $I_1(U_k) = I(W_1;Y) - I(W_1;U_k)$ and $I_2(U_k) = I(W_2;Y,W_1) - I(W_2;U_k)$. This figure illustrates a possible achievable rates region.

Remark 1 *By including the time sharing random variable, it can be easily verified that linear codes achieve the capacity of the standard MAC. Furthermore, if $H(W_1|Y) = H(W_2|Y,W_1)$, the region $\mathcal{R}_1(a,b)$ is a pentagon.*

3.3 Sequential Decoding via Compute-and-Forward Technique

For the case of 2 input codewords the decoder consists of 2 steps. The first one is decoding the modulo sum of the codewords and the second one is decoding one of the codewords using the sum as a side information. The decoder is based on *Maximum A Posteriori* (MAP) rule which in the case of equally likely *a priori* distribution becomes *Maximum Likelihood* (ML).

1^{st} Step:

$$\hat{s}_i = \underset{s_i}{\operatorname{argmax}}\, p(s_i|y^n) \tag{3.4}$$

2^{nd} Step:

$$\hat{x}_{1i} = \underset{x_{1i}}{\operatorname{argmax}}\, p(x_{1i}|y^n, s^n) \tag{3.5}$$

The equation (3.4) and (3.5) are the general MAP rules [1]. However, our main focus will remain binary linear codes with codebook $\mathcal{C} \subseteq \mathbb{F}_2^n$ ($q = 2$ in this case). An useful notation is the set $\{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$ denoted by $\sim x_i$. Also H is an $(n-k) \times n$ matrix and $H = [f_1, f_2, \ldots, f_{n-k}]^\top$. Therefore the procedure will be as follows:

1^{st} Step:

$$\begin{aligned}
\hat{s}_i &= \underset{s_i \in \{0,1\}}{\operatorname{argmax}}\, p(s_i|y^n) \\
&= \underset{s_i \in \{0,1\}}{\operatorname{argmax}} \sum_{\sim s_i} p(s^n|y^n) \\
&= \underset{s_i \in \{0,1\}}{\operatorname{argmax}} \sum_{\sim s_i} \frac{p(s^n, y^n)}{p(y^n)} \\
&= \underset{s_i \in \{0,1\}}{\operatorname{argmax}} \sum_{\sim s_i} p(y^n|s^n)p(s^n) \\
&= \underset{s_i \in \{0,1\}}{\operatorname{argmax}} \sum_{\sim s_i} \prod_{i=1}^{n} p(y_i|s_i) \frac{1}{|\mathcal{C}|} \mathbb{1}_{\{sH^\top = 0\}} \\
&= \underset{s_i \in \{0,1\}}{\operatorname{argmax}} \sum_{\sim s_i} \left(\prod_{i=1}^{n} p(y_i|s_i) \frac{1}{|\mathcal{C}|} \prod_{j=1}^{n-k} \mathbb{1}_{\{sf_j^\top = \mathbf{0}\}} \right)
\end{aligned} \tag{3.6}$$

[1] For the sake of simplicity both u and x are represented as x because $x = f(u)$ and in addition $p(y|x) = p(y|u)$. For this reason $x_{1i} \in \{0,1\}$ is used instead of $x_{1i} \in \{-1,1\}$ (3.7)

2^{nd} Step:

$$\begin{aligned}
\hat{x}_{1i} &= \underset{x_{1i}\in\{0,1\}}{\operatorname{argmax}}\, p(x_{1i}|y^n, s^n) \\
&= \underset{x_{1i}\in\{0,1\}}{\operatorname{argmax}} \sum_{\sim x_{1i}} p(x_1^n|y^n, s^n) \\
&= \underset{x_{1i}\in\{0,1\}}{\operatorname{argmax}} \sum_{\sim x_{1i}} \frac{p(x_1^n, y^n, s^n)}{p(y^n, s^n)} \\
&= \underset{x_{1i}\in\{0,1\}}{\operatorname{argmax}} \sum_{\sim x_{1i}} p(y^n|x_1^n, s^n)p(s^n|x_1^n)p(x_1^n) \qquad (3.7) \\
&= \underset{x_{1i}\in\{0,1\}}{\operatorname{argmax}} \sum_{\sim x_{1i}} \prod_{i=1}^{n} p(y_i|x_{1i}, s_i)\frac{1}{|\mathcal{C}|^2}\mathbb{1}_{\{x_1 H^\top=0\}} \\
&= \underset{x_{1i}\in\{0,1\}}{\operatorname{argmax}} \sum_{\sim x_{1i}} \left(\prod_{i=1}^{n} p(y_i|x_{1i}, s_i)\frac{1}{|\mathcal{C}|^2}\prod_{j=1}^{n-k}\mathbb{1}_{\{xf_j^\top=\mathbf{0}\}}\right)
\end{aligned}$$

Note 1 *Assuming a uniform distribution over the codewords of codebook* $\mathcal{C}$

$$p(x_1^n) = \begin{cases} \frac{1}{|\mathcal{C}|} & x_1^n \text{ is a codeword} \\ 0 & otherwise \end{cases} = \frac{1}{|\mathcal{C}|}\mathbb{1}_{\{x_1 H^\top=0\}} \qquad (3.8)$$

$$\begin{aligned}
p(s^n|x_1^n) &= \mathbb{P}(S^n = s^n|X_1^n = x_1^n) \\
&= \mathbb{P}(X_2^n = s^n \oplus x_1^n|X_1^n = x_1^n) = \frac{1}{|\mathcal{C}|}
\end{aligned} \qquad (3.9)$$

The term x_1^n *is already a codeword due to the fact that if not, the term* $p(s^n|x_1^n)p(x_1^n)$ *is already 0. Also* s^n *is a codeword because it is the outcome of the first step (SPA) which in every case even if wrong decoded it outputs a codeword. Therefore, it is left that* x_2^n *is already a codeword and for this reason equation* (3.9) *contains no indicator function.*

Note 2 *As we have assume a Gaussian channel, we can write the Probability Density Function* (PDF) *of the channel as follows:*

$$p(y_i|x_{1i}, x_{2i}) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(y_i - x_{1i} - x_{2i}\right)^2\right)$$

The only term not specified explicitly remains, in [14]:

$$\begin{aligned}
p(y_i|s_i) &= \sum_{x_{1i}} p(y_i|x_{1i}, s_i)p(x_{1i}|s_i) \\
&= \frac{1}{2}\sum_{x_{1i}} p(y_i|x_{1i}, x_{2i} = s_i \oplus x_{1i})
\end{aligned}$$

Our next task is reduced into converting SD via CF steps into factor graphs. Surprisingly, each step of SD can be represented by a factor graph containing only a single tanner graph. This statement is supported by equations (3.6) and (3.7) which contain a single indicative function. In other words the graph will be determined only by the parity check matrix H and LLR-s.

Example 3.3.1 *The corresponding factor graph of the first step of SD via CF is illustrated by this example.*

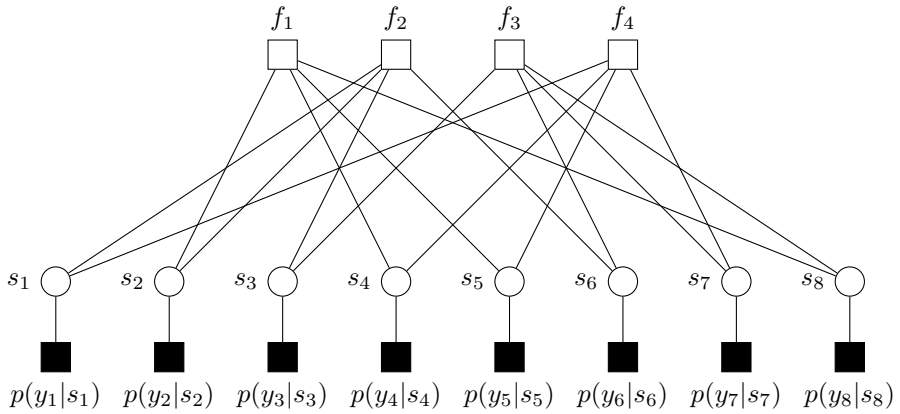

Figure 3.3: Factor graph corresponding to the parity check matrix in equation (2.3) and input probabilities which are in LLR form.

$$\begin{aligned} LLR_1 &= \log \frac{p(y_i|s_i=0)}{p(y_i|s_i=1)} \\ &= \log\left(\frac{\sum_{x_{1i}} p(y_i|x_{1i}, x_{2i}=x_{1i})}{\sum_{x_{1i}} p(y_i|x_{1i}, x_{2i}=1\oplus x_{1i})}\right) \\ &= \log\left(\frac{e^{-\frac{1}{2}(y_i-2\sqrt{P})^2}+e^{-\frac{1}{2}(y_i+2\sqrt{P})^2}}{2e^{-\frac{1}{2}y_i^2}}\right) \end{aligned} \tag{3.10}$$

$$\begin{aligned} LLR_2 &= \log \frac{p(y_i|x_{1i}=0, s_i)}{p(y_i|x_{1i}=1, s_i)} \\ &= \log\left(\frac{p(y_i|x_{1i}=0, x_{2i}=s_i)}{p(y_i|x_{1i}=1, x_{2i}=s_i\oplus 1)}\right) \\ &= \begin{cases} 4y_i\sqrt{P} & \text{for } s_i=0 \\ 0 & \text{for } s_i=1 \end{cases} \end{aligned} \tag{3.11}$$

3.3.1 Implementation of LLR-s

Clearly, a direct observation which is not favourable for us is the complicated expression of LLR_1 containing exponential expression. Therefore, we have to come up with a simplification method. In the vast majority of literature the log-sum of exponential terms is defined as:

$$\max^*(x, y) = \log(e^x + e^y) \tag{3.12}$$

which is shown in [15] to be

$$\max^*(x, y) = \max(x, y) + \log(1 + e^{-|x-y|}) \tag{3.13}$$

Apparently, the function depends on the difference $\Delta = x - y$ and the term containing it serves as an *correction term*. Due to this reason, a linear approximation with respect to the difference Δ will be done. The function will have the following look-up table:

$\Delta = x - y$	$\max^*(x, y)$
$>$ 2.507	y
$<$ -2.507	x
(0, 2.507]	$y - 2.49(\Delta - 2.507)$
(-2.507, 0]	$y + 2.49(\Delta + 2.507)$

Table 3.1: Linear Approximation of the auxiliary function $\max^*(x, y)$ for different values of $\Delta = x - y$

3.4 Achieving boundary points on capacity region

After we have defined the achievable rates region in section 3.2 now we are ready to aim for some of those boundary points which are assumed to be the most difficult points to achieve. The theoretical capacity calculation is done for binary input symbols (not for gaussian) which would be a fair comparison for our case. Moreover, the actual comparison is between the theoretical SNR to achieve error free transmission and the actual SNR to achieve *Error Probability* or *Bit Error Rate* (BER) at 10^{-4}.

The SNR is represented by P_k/σ_w^2, which is the user power P_k over the gaussian noise variance σ_w^2. Throughout simulations noise variance σ_w^2 is set to 1 to keep the calculations simple.

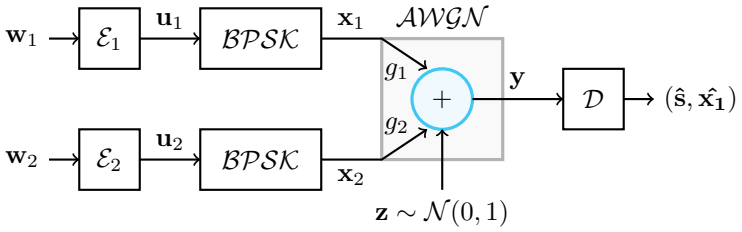

Figure 3.4: Multiple access communication system specifications.

- Encoders($\mathcal{E}_1, \mathcal{E}_2$): $u_1 = w_1 G \oplus d_1$ where G is the generator matrix and d is the *dither* vector.
- Modulator($\mathcal{BPSK}$): $x_{1i} = \sqrt{P}(1-2u_{1i})$ where P is the power assigned to the first user. This mapping is called *Binary Phase Shift Keying* (BPSK).
- Channel($\mathcal{AWGN}$): $y = g_1 x_1 + g_2 x_2 + z$ where g_1, g_2 are the channel gains and z is the gaussian noise of zero mean and unity variance. Since having different channel gains decoding one of the codewords would be easier than to decode the other one we focus on the special case $(g_1, g_2) = (1, 1)$ and the channel equation becomes $y = x_1 + x_2 + z$.
- Decoder($\mathcal{D}$): The method used by the decoder is SD via CF. At the end the decoder should be able to decode both codewords separately $(\hat{x_1}, \hat{x_2})$ by firstly decoding sum $\hat{s} = (x_1 \hat{\oplus} x_2)$ and then one of the codewords $\hat{x_i}$ where $i \in \{1, 2\}$.

3.4.1 First Scenario: Symmetric Rate

In this scenario the power for the user 1 and user 2 is the same $P_1 = P_2$. The aimed point is the symmetric rate pair $(R_1, R_2) = (\frac{1}{3}, \frac{1}{3})$ which is shown in figure 3.5. The corresponding theoretical SNR is 0.097dB under the condition of using binary input symbols. To calculate the actual SNR we will use SPA algorithm as in [16] embedded perfectly in SD via CF. If the modulo sum

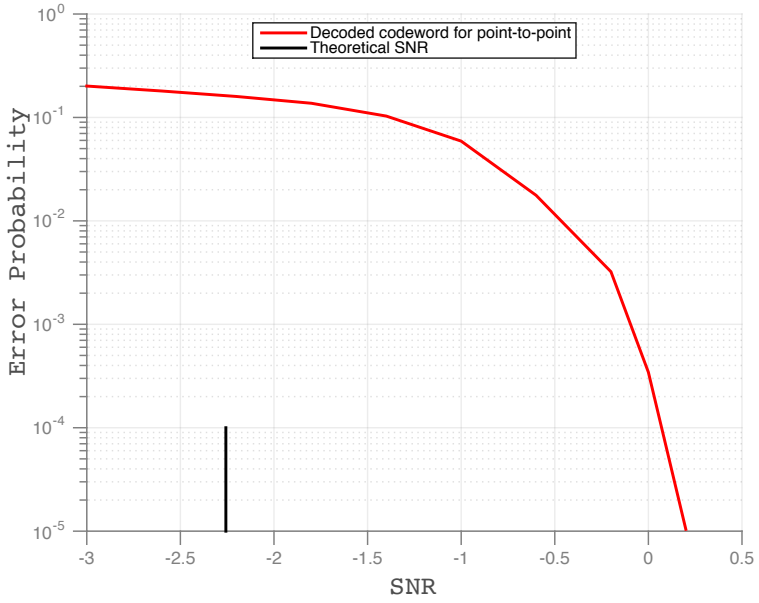

FIGURE 3.5: Simulation results for $\frac{1}{3}$ rate code that can achieve within 2.25 dB of the Shannon limit.

of the users is successfully decoded then we have no error propagation. An important aspect used throughout all the simulations is *Coded Modulation Library* (CML) package. The CML is a library of communication system simulations developed at West Virginia University. Also it is an open source toolbox for simulating capacity approaching codes in Matlab. It also uses C-mex to reduce the complexity (running time). Even thought CML has many features and ready supported standards (UMTS, CCSDS, DVB-RCS, WiMAX IEEE 802.16, DVB-S2 etc) we used only a few ingredients.

The LDPC codes used here are taken from Mackay's site [17] of block length 1920. Mackay himself has created a whole database of Gallager's codes containing performance results for the point-to-point communication. In figure 3.6 simulation results for $\frac{1}{3}$ rate code and point-to-point case are shown. Moreover, this LDPC code can achieve within 2.25 dB of the Shannon limit. This result is particularly surprising since such codes are extremely easy to implement. Using the same code as above for the 2-user Gaussian MAC to decode the pair $(R_1, R_2) = (\frac{1}{3}, \frac{1}{3})$ the actual SNR for achieving *Error Probability* around 10^{-4} is approximately 1.75 dB as shown in figure 3.7. From another perspective, we are 1.653 dB away from the Shannon limit. Should someone consider it as a reasonable result? Taking into consideration the fact that the code itself has 2.25 dB loss, the simulation results we are obtaining from sequential decoding via CF are more than reasonable. While this may not be entirely rigorous, it infers that the code we are using is far from perfect.

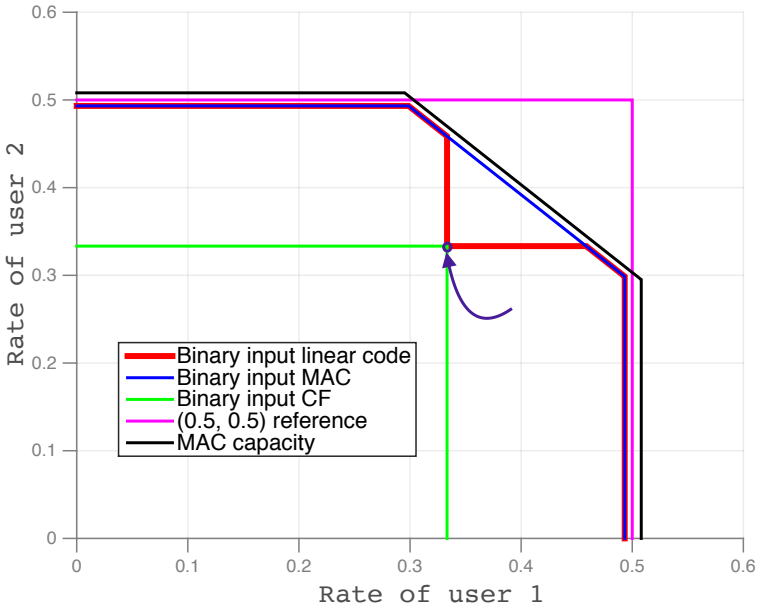

Figure 3.6: Target Point corresponding to rate pair $(R_1, R_2) = (\frac{1}{3}, \frac{1}{3})$ at theoretical SNR of 0.097 dB.

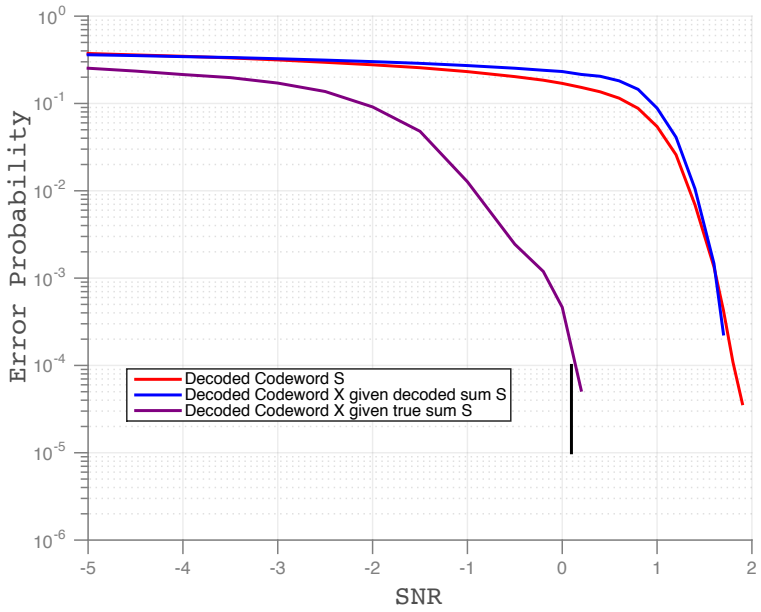

Figure 3.7: Approximately 1.653 dB away from the Shannon limit for the symmetric rate pair $(R_1, R_2) = (\frac{1}{3}, \frac{1}{3})$.

3.4.2 Second Scenario: Asymmetric Rate

In this scenario the power for the user 1 and user 2 is the same $P_1 = P_2$. The aimed point is the asymmetric rate pair $(R_1, R_2) = (\frac{9}{20}, \frac{1}{3})$ which is shown in figure 3.8. The corresponding theoretical SNR is 0.097dB under the condition of using binary input symbols. Surprisingly, the theoretical SNR is the same for symmetric rate pair. To calculate the actual SNR we should construct nested linear codes whose performance does not degrade at high rates. Based on rate $\frac{1}{3}$ code which have a reasonably good performance we want to construct rate $\frac{9}{20}$ code by adapting some of the check nodes of the rate $\frac{1}{3}$ code. The relation between codebooks is: $\mathcal{C}_{\frac{1}{3}} \subseteq \mathcal{C}_{\frac{9}{20}}$

One of the possible strategies is by removing some check nodes of the rate $\frac{1}{3}$ code, however the performance is degraded by a lot of margin. Another strategy is by "merging" some of the check nodes into a single one. Explaining "merging" technique into mathematical expression: $f_k = f_i \oplus f_j$ where $k \in \{1, 2, \ldots, 1056\}$ and $i, j \in \{1, 2, \ldots, 1280\}$. While this is not our main purpose it is a key ingredient of simulation results. Moreover the performance of hybrid rate $\frac{9}{20}$ code is very close and even better than the performance of rate $\frac{1}{3}$ code. Having a slightly better performance is actually intuitive and we can indeed increase this margin, however it become a problem of designing adaptive rate code (not in our scope). The rate adaptive technique used is arbitrarily chosen, however the performance is roughly stable for different simulations.

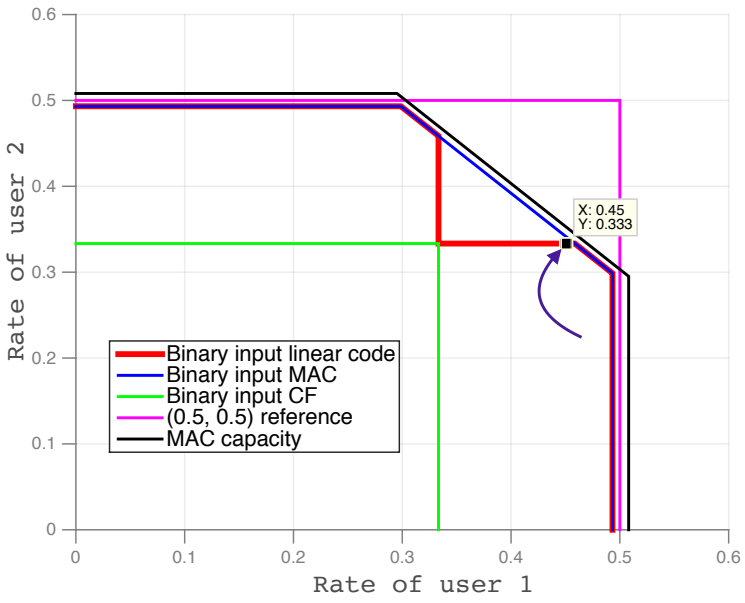

FIGURE 3.8: Target Point corresponding to rate pair $(R_1, R_2) = (\frac{9}{20}, \frac{1}{3})$ at theoretical SNR of 0.097 dB.

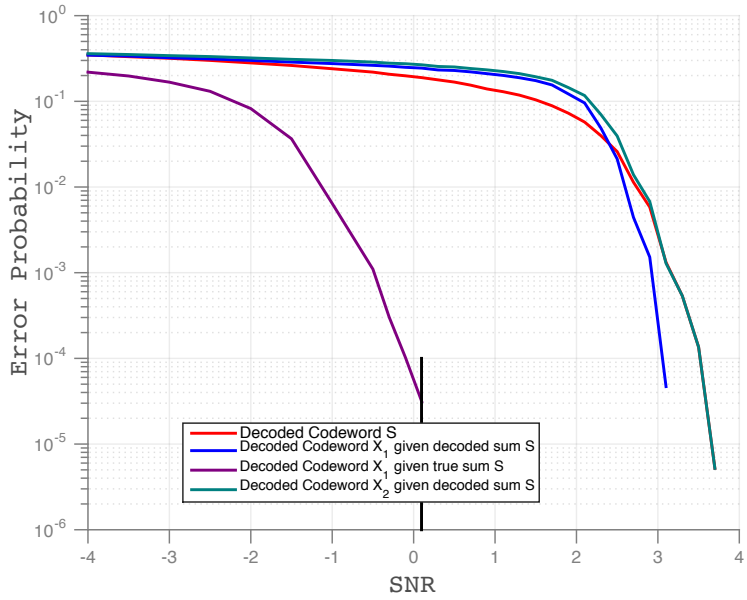

Figure 3.9: Approximately 2.95 dB away from Shannon limit for codeword$_1$ and 3.4 dB for codeword$_2$ in case of asymmetric rate pair $(R_1, R_2) = (\frac{9}{20}, \frac{1}{3})$.

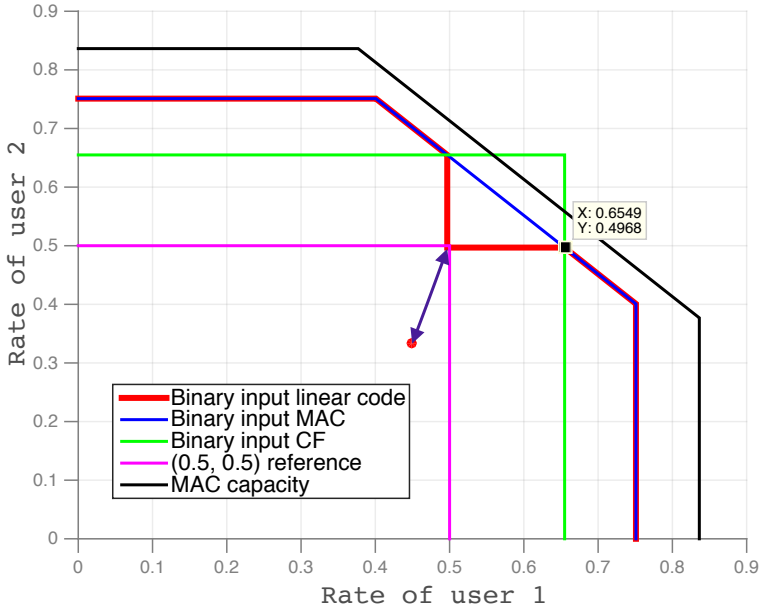

Figure 3.10: Rate loss for user$_1$ is 0.0468 and rate loss for user$_2$ is 0.1635 in case of asymmetric rate pair $(R_1, R_2) = (\frac{9}{20}, \frac{1}{3})$.

In terms of results in order to decode the rate pair $(R_1, R_2) = (\frac{9}{20}, \frac{1}{3})$ the actual SNR for achieving *Error Probability* around 10^{-4} is approximately 3.05 dB for codeword$_1$ and 3.5 dB for codeword$_2$ as shown in figure 3.9. In other words, we are 2.95 dB away from the Shannon limit for codeword$_1$ and 3.4 dB away for codeword$_2$. Let us look from another perspective and analyse the rate loss. Our goal is to achieve the boundary point, but where does the point fall into the achievable rates region. According to the simulation results user$_1$ has 0.0468 rate loss (or away from the achievable rates limit) and user$_2$ has 0.1635 rate loss as it is depicted in figure 3.10.

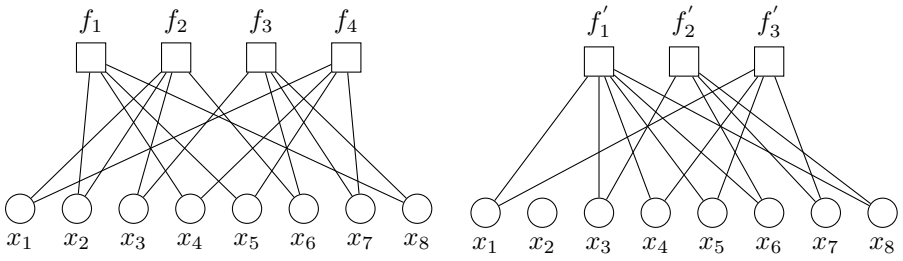

FIGURE 3.11: Example which illustrates how to adapt the rate of a code. (a) sample code, (b) rate adapted code.

Considering example 3.11 the adapted check nodes have thee following form: $f_1^{'} = f_1 \oplus f_2$, $f_2^{'} = f_2$ and $f_3^{'} = f_3$. The above example illustrates how to increase the rate of a code from $\frac{1}{2}$ to $\frac{5}{8}$. Apparently, the example 3.11 that I choose on purpose lacks connectivity. The node corresponding to 2^{nd} bit of the codeword is not connected and SPA algorithm does not update this node, which may lead to a permanent error. To overcome this issue we must make sure that the check nodes (forming adapted check nodes) have disjoint set of variable nodes. In our case $\{x_2, x_4, x_5, x_8\} \cap \{x_1, x_2, x_3, x_6\} = \{x_2\}$. Loosely speaking the problem above is solved somehow by itself due to low density structure of LDPC codes. The parity check matrix we are using has dimension 1280×1920 where each column has a weight of 2 or 3. Due to the sparseness of the code the probability to encounter such a problem is quite small. A rigorous way to verify such a problem is to check whether the number of 1-s in the sample parity check is equal to the number of 1-s in the adapted parity check.

3.5 Extension to 4-PAM

The only step which changes from before is the modulation block. One of the limitations is that achievable rate region leaves aside all the points whenever both rates are grater than $\frac{1}{2}$ at the same time. To overcome this issue we will use 4-PAM modulation instead of BPSK. One could suggest to use a higher order modulation, however this leads to significantly high dB loss, degrading the performance.

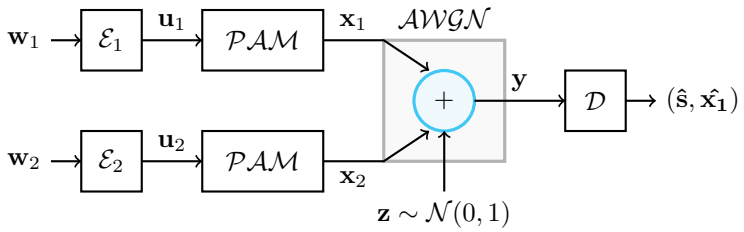

Figure 3.12: Multiple access communication system specifications for 4-PAM modulation.

- Modulator($\mathcal{PAM}$): $x_{1i} = \frac{\sqrt{P}}{5}\left((2u_{1i}^1 - 1)\cdot 2 + (2u_{1i}^0 - 1)\right)$ where P is the power assigned to the first user and the binary representation for u_1 is $u_1^1 u_1^0$. This mapping is called 4-PAM. The mapping can be done in 24 different ways and the one we shown before is the simplest one. Without taking into consideration the power scale it maps $00 \to -3$, $01 \to -1$, $10 \to 1$ and $11 \to 3$.

The decoding SD via CF method will remain the same as before except the input LLR-s. Intuitively the number of LLR (which should be in vector form due to 4 possible values) is halved taking into account the fact that 2 bits are mapped into 1 symbol. To overcome this issue we split LLR into 2 part for each of the SD via CF steps as below:

$$\begin{aligned} LLR_1^1 &= \log\left(\frac{\sum_{s_i^1} p(y_i|s_i^1 s_i^0 = s_i^1 0)}{\sum_{s_i^1} p(y_i|s_i^1 s_i^0 = s_i^1 1)}\right) \\ LLR_1^2 &= \log\left(\frac{\sum_{s_i^0} p(y_i|s_i^1 s_i^0 = 0 s_i^0)}{\sum_{s_i^0} p(y_i|s_i^1 s_i^0 = 1 s_i^0)}\right) \end{aligned} \tag{3.14}$$

$$\begin{aligned} LLR_2^1 &= \log\left(\frac{\sum_{x_{1i}^1} p(y_i|x_{1i}^1 x_{1i}^0 = x_{1i}^1 0, s_i^1 s_i^0)}{\sum_{x_{1i}^1} p(y_i|x_{1i}^1 x_{1i}^0 = x_{1i}^1 1, s_i^1 s_i^0)}\right) \\ LLR_2^2 &= \log\left(\frac{\sum_{x_{1i}^0} p(y_i|x_{1i}^1 x_{1i}^0 = 0 x_{1i}^0, s_i^1 s_i^0)}{\sum_{x_{1i}^0} p(y_i|x_{1i}^1 x_{1i}^0 = 1 x_{1i}^0, s_i^1 s_i^0)}\right) \end{aligned} \tag{3.15}$$

In this scenario again the power for the user 1 and user 2 is the same $P_1 = P_2$. The aimed point is the asymmetric rate pair $(R_1, R_2) = (\frac{9}{10}, \frac{2}{3})$ which is shown in figure 3.13. The corresponding theoretical SNR is 6.154dB under the condition of using quaternary input symbols. To calculate the actual SNR we use the constructed nested linear codes from before. From simulation results in order to decode rate the pair $(R_1, R_2) = (\frac{9}{10}, \frac{2}{3})$ the actual SNR for achieving *Error Probability* around 10^{-4} is approximately 5.5 dB away form the Shannon limit for codeword$_1$ and 5.65 dB for codeword$_2$ as shown in figure 3.14.

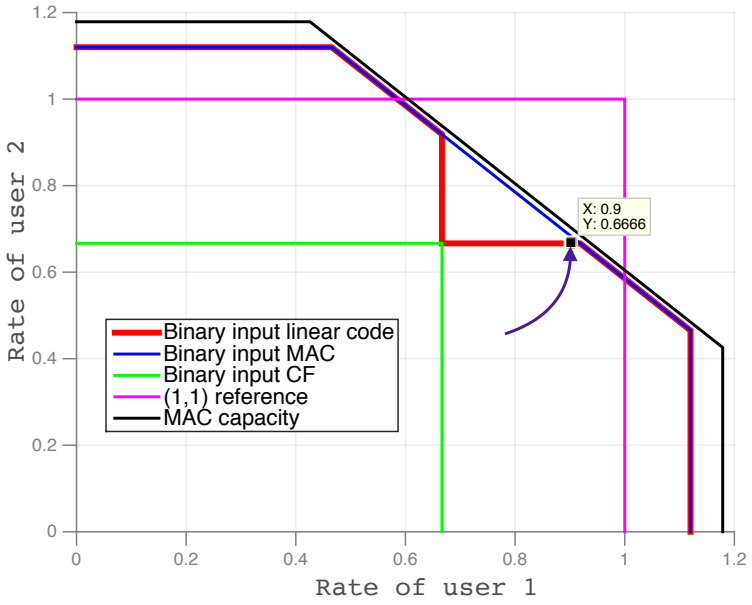

FIGURE 3.13: Target Point corresponding to rate pair $(R_1, R_2) = (\frac{9}{10}, \frac{2}{3})$ at theoretical SNR of 6.154 dB.

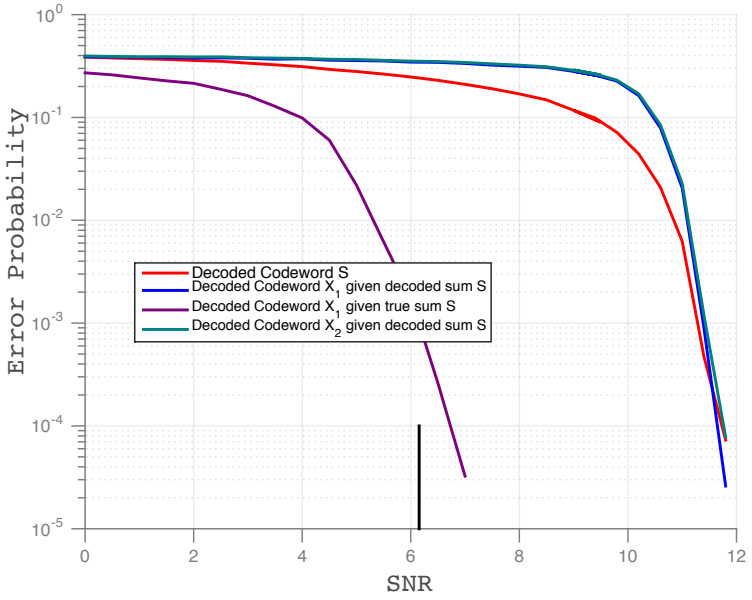

FIGURE 3.14: Approximately 5.5 dB away from Shannon limit for codeword$_1$ and 5.65 dB for codeword$_2$ in case of asymmetric rate pair $(R_1, R_2) = (\frac{9}{10}, \frac{2}{3})$

3.6 Extension to Multiple Users

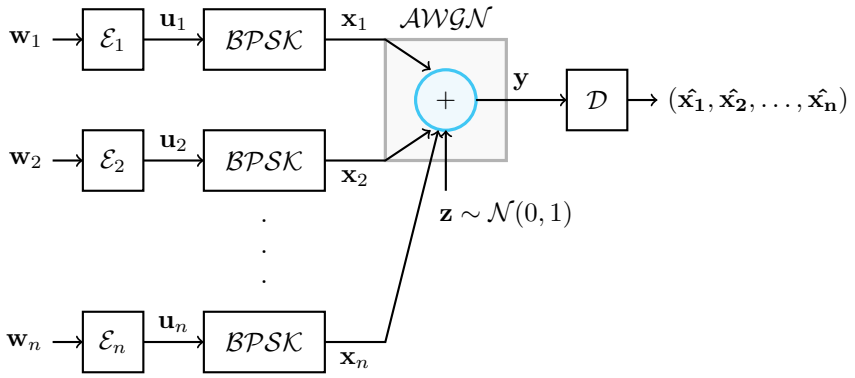

Figure 3.15: Multiple access communication system specifications for multiple users.

- Channel($\mathcal{AWGN}$): $y = x_1 + x_2 + \cdots + x_n + z$ where z is the gaussian noise of zero mean and unity variance. The equation is a special case when channel gains are $(g_1, g_2, \ldots, g_n) = (1, 1, \ldots, 1)$.

For the case of k input signals the decoder consists of k steps. Let us introduce a new parameter needed throughout SD via CF method: $w_k = x_1 \oplus x_2 \oplus \cdots \oplus x_k$, $k \in \{1, 2, \ldots, n\}$, with initial value $w_1 = x_1$. The first step is to decode the modulo sum of the codewords and the other steps are decoding parameter w_{k-1} using w_k as a side information. The decoder is based on MAP rule, which in this case is simply ML.

1^{st}*Step*:

$$\hat{w}_{in} = \underset{w_{in}}{\operatorname{argmax}}\, p(w_{in}|y^n) \tag{3.16}$$

2^{nd}*Step*:

$$\hat{w}_{i(n-1)} = \underset{w_{i(n-1)}}{\operatorname{argmax}}\, p(w_{i(n-1)}|y^n, w_n^n) \tag{3.17}$$

.
.
.

n^{th}*Step*:

$$\hat{w}_{i1} = \underset{w_{i1}}{\operatorname{argmax}}\, p(w_{i1}|y^n, w_n^n, w_{n-1}^n, \ldots, w_2^n) \tag{3.18}$$

The equation (3.16), (3.17) and (3.18) are the general MAP rules. An useful notation is the set $\{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$ denoted by $\sim x_i$. Also H is $(n-k) \times n$ matrix and $H = [f_1, f_2, \ldots, f_{n-k}]^\top$. The procedure for the kth step will be as follows:

k^{th} *Step*:

$$\begin{aligned}
\hat{w}_{ik} &= \underset{w_{ik}\in\{0,1\}}{\operatorname{argmax}}\, p(w_{ik}|y^n, w_n^n, w_{n-1}^n, \ldots, w_{k+1}^n) \\
&= \underset{w_{ik}\in\{0,1\}}{\operatorname{argmax}} \sum_{\sim w_{ik}} p(w_k^n|y^n, w_n^n, w_{n-1}^n, \ldots, w_{k+1}^n) \\
&= \underset{w_{ik}\in\{0,1\}}{\operatorname{argmax}} \sum_{\sim w_{ik}} \frac{p(w_k^n, y^n, w_n^n, w_{n-1}^n, \ldots, w_{k+1}^n)}{p(y^n, w_n^n, w_{n-1}^n, \ldots, w_{k+1}^n)} \\
&= \underset{w_{ik}\in\{0,1\}}{\operatorname{argmax}} \sum_{\sim w_{ik}} p(y^n|w_n^n, w_{n-1}^n, \ldots, w_k^n) p(w_n^n, w_{n-1}^n, \ldots, w_k^n) \qquad (3.19) \\
&\overset{(*)}{=} \underset{w_{ik}\in\{0,1\}}{\operatorname{argmax}} \sum_{\sim w_{ik}} \prod_{i=1}^{n} p(y_i|w_{in}, \ldots, w_{ik}) \frac{1}{|\mathcal{C}|^{n-k+1}} \mathbb{1}_{\{w_k H^\top = 0\}} \\
&= \underset{w_{ik}\in\{0,1\}}{\operatorname{argmax}} \sum_{\sim w_{ik}} \left(\prod_{i=1}^{n} p(y_i|w_{in}, \ldots, w_{ik}) \frac{1}{|\mathcal{C}|^{n-k+1}} \prod_{j=1}^{n-k} \mathbb{1}_{\{w_k f_j^\top = \mathbf{0}\}} \right)
\end{aligned}$$

Note 3 *Assuming a uniform distribution over the codewords of codebook* $\mathcal{C}$

$$p(w_k^n) = \begin{cases} \frac{1}{|\mathcal{C}|} & w_k^n \text{ is a codeword} \\ 0 & \text{otherwise} \end{cases} = \frac{1}{|\mathcal{C}|} \mathbb{1}_{\{w_k H^\top = 0\}} \qquad (3.20)$$

$$\begin{aligned}
p(w_m^n|w_{m-1}^n, \ldots, w_k^n) &= \mathbb{P}(W_m^n = w_m^n | W_{m-1}^n = w_{m-1}^n, \ldots, W_k^n = w_k^n) \\
&= \mathbb{P}(X_m^n = w_m^n \oplus w_{m-1}^n | W_{m-1}^n = w_{m-1}^n, \ldots, W_k^n = w_k^n) \\
&= \mathbb{P}(X_m^n = w_m^n \oplus w_{m-1}^n) = \frac{1}{|\mathcal{C}|}
\end{aligned} \qquad (3.21)$$

The terms w_m^n, w_{m-1}^n, $\ldots$, w_{k+1}^n *are already codewords as a result of previous steps which in every case, even if wrong decoded, it outputs a codeword. Also* w_k^n *is a codeword due to the fact that if not, the term* $p(w_m^n, w_{m-1}^n, \ldots | w_k^n) p(w_k^n)$ *is already 0. As a result, we are left with* x_m^n *is already a codeword and for this reason equation* (3.21) *contains no indicator function. The equation* $(*)$ *is proved by using the chain rule together with equation* (3.20) *and* (3.21) *for* $m \in \{k+1, k+2, \ldots, n\}$*:*

$$\begin{aligned}
p(w_n^n, w_{n-1}^n, \ldots, w_k^n) &= p(w_k^n) \prod_{m=k+1}^{n} p(w_m^n | w_{m-1}^n, \ldots, w_k^n) \\
&= \frac{1}{|\mathcal{C}|} \mathbb{1}_{\{w_k H^\top = 0\}} \cdot \frac{1}{|\mathcal{C}|^{n-k}}
\end{aligned}$$

3.7 Asymmetric Channel Gain

Throughout the thesis we have focused on symmetric channel gain, in particular channel gain pair $(g_1, g_2) = (1, 1)$. However, asymmetric channel gain can be of a particular importance. To support the previous statement, the motivation comes from the fact that channels in reality are never symmetric due to many contributing factors. In this scenario the power for the user-1 and user-2 is the same $P_1 = P_2$. The aimed point is the symmetric rate pair $(R_1, R_2) = (\frac{1}{3}, \frac{1}{3})$. Considering sequential decoding via CF, in the second step we will decode the codeword corresponding to the greater channel gain. For all simulation results in table 3.2 we are looking for the SNR value corresponding to *Error Probability* 10^{-4} as depicted in figures 3.16, 3.17 and 3.18. As the gap between channel gains increases the performance improves and this is intuitively expected.

Channel gain pair	Theoretical SNR	Practical SNR	Δ-SNR
$(1,\sqrt{2})$	-1.16	0.4	1.56
$(1,\sqrt{3})$	-1.67	-0.3	1.37
(1,2)	-1.92	-1.25	0.67

Table 3.2

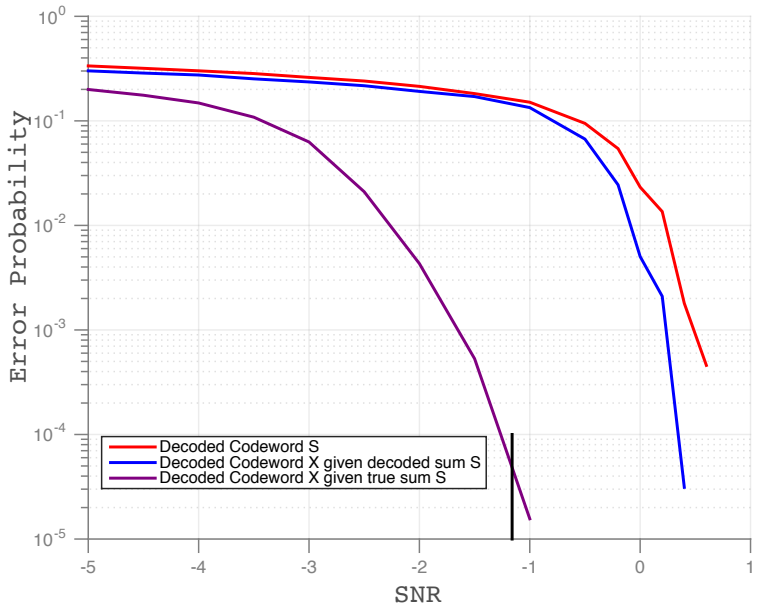

Figure 3.16: Simulation results for channel gain pair $(g_1, g_2) = (1, \sqrt{2})$.

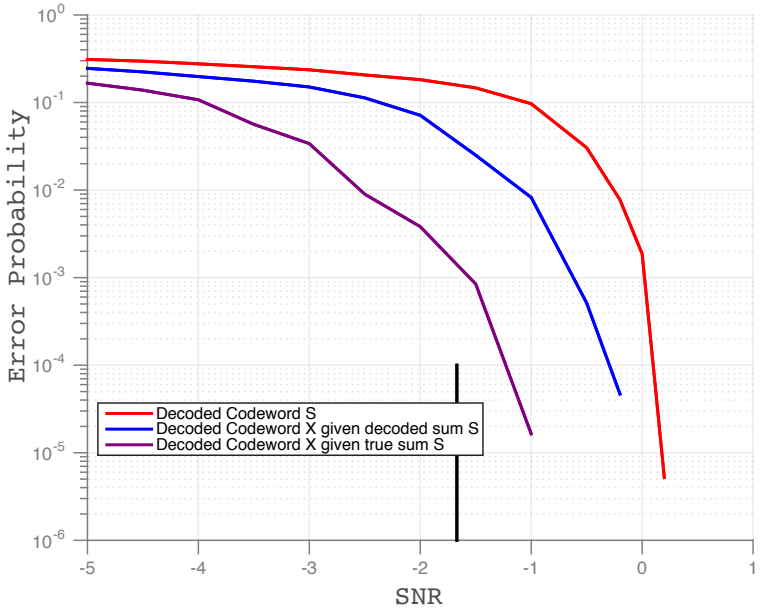

FIGURE 3.17: Simulation results for channel gain pair $(g_1, g_2) = (1, \sqrt{3})$.

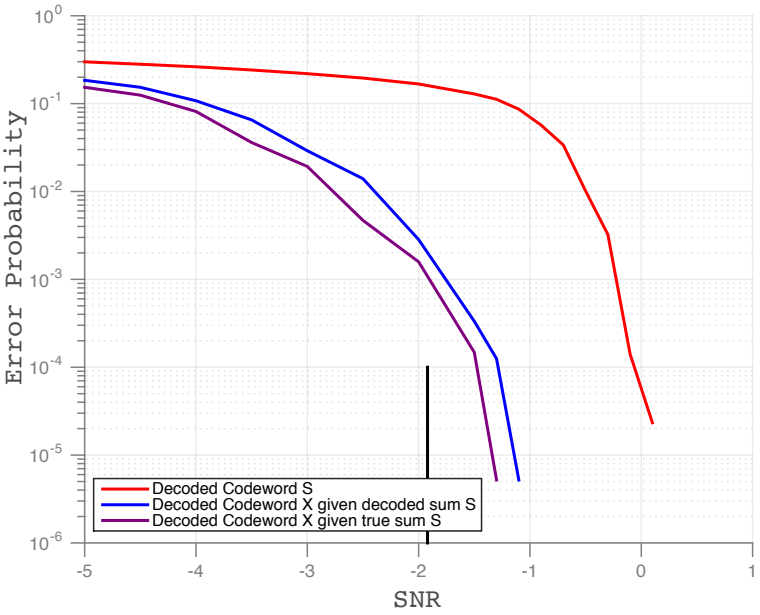

FIGURE 3.18: Simulation results for channel gain pair $(g_1, g_2) = (1, 2)$.

Conclusion 4

For the Gaussian MAC, we have shown that sequential decoding via compute-and-forward technique is able to achieve boundary points on the capacity region with reasonable complexity and good performance. A set of practical LDPC codes with moderate block length are used to investigate the practical performance of the proposed technique. In order to match the BER performance under the BPSK modulation, the binary input channel capacity equation is used to obtain the theoretical values as the benchmark for the simulations. For this particular case of BPSK modulation, under the conditions of equal channel gains and equal power of users, the capacity region leaves aside all the rate pairs of users that are greater than $\frac{1}{2}$ at the same time. To overcome this issue another modulation named $4-$PAM is used to achieve rate pairs within $(0, 1)$ interval that are not achieved using BPSK modulation.

Surprisingly, simulation results show that sometimes multiples access surpasses the performance of point-to-point communication. From modulation perspective, $4-$PAM scenario degrades the performance over BPSK scenario, but it still remains a key ingredient of our thesis.

Bibliography

[1] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, February 2001.

[2] B. Rimoldi and R. Urbanke, "A rate-splitting approach to the gaussian multiple-access channel," *IEEE Transactions on Information Theory*, vol. 42, no. 2, March 1996.

[3] B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference through structured codes," *IEEE Transactions on Information Theory*, October 2011.

[4] J. Zhu and M. Gastpar, "Multiple access via compute-and-forward," *IEEE*, July 2014.

[5] A. Betten, M. Brown, and H. Fripertinger, *Error-Correcting Linear Codes*. Springer, July 2006, vol. 18.

[6] R. Zamir, S. Shamai, and U. Erez, "Nested linear/lattice codes for structured multiterminal binning," *IEEE Transactions on Information Theory*, vol. 57, pp. 1250 – 1276, June 2002.

[7] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, September 1981.

[8] R. G. Gallager, *Low Density Parity Check Codes*. M.I.T. Press, July 1963.

[9] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, February 2001.

[10] W. E.Ryan, *An Introduction to LDPC Codes*, August 2003.

[11] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[12] A. E. Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge University Press, December 2011.

[13] S. H. Lim and M. Gastpar, "Coding theorems via linear codes: Joint decoding rate regions," *IEEE*, 2015.

[14] J. Zhu and M. Gastpar, "Compute-and-forward using nested linear codes for the gaussian mac," *IEEE*, pp. 1–5, May 2015.

[15] A. J. Viterbi, "An intuitive justification and a simplified implementation of the map decoder for convolutional codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 260–264, February 1998.

[16] S. Lin and D. J. Costello, *Error Control Coding.* Pearson Prentice Hall, 2004.

[17] (2002, April). [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html