Aspects of Density Evolution in LDPC Key Reconciliation

Bachelor Thesis

Yilun Zhang
Electrical and Computer Engineering
Jacobs University Bremen
yi.zhang@jacobs-university.de
Advisor: Prof.Dr.-Ing.W. Henkel, N. Islam

June 1, 2015

Abstract

With increasing significance of wireless transmission, physical layer security becomes a challenging topic resulting from the Shannon's limit. This work explores physical layer security from the aspect of information theory and discusses key generation and reconciliation by Slepian-Wolf coding and low-density parity-check (LDPC) code. In order to design the LDPC code, we use density evolution on the log-likelihood ratio (LLR) distribution which is forwarded along the edges in Belief-Propagation (BP) algorithm. This paper focuses on the density evolution procedure when the measured distribution is non-consistent and non-Gaussian and follows the iterations step by step. Eventually, the research finds that though the initial measurement in Bob's side is obviously non-Gaussian and non-consistent, after several iterations or even only convolutions with the densities from other nodes within an iteration, a predictive result occurs. Though the final result in this work is still not Gaussian but the finding could say that the distribution tends to be consistent Gaussian. This result will contribute to the further studies of physical layer security.

Contents

1 Introduction **1**

2 Background **2**
2.1 Shannon theory and Perfect Secrecy 2
2.2 Key generation and reconciliation by Slepian-Wolf coding 4
2.3 LDPC codes 5
2.4 Belief-Propagation Decoding 6
2.5 Density Evolution 8

3 Calculation and Simulation **11**
3.1 Decoding at a Variable Node 11
3.2 Decoding at a Check Node 11
3.3 Calculation and Simulation 13
3.3.1 General Operation 13
3.3.2 Calculation in the Complex Domain 17
3.3.3 2D Convolution 21
3.3.4 1D calculation and simulation 32

4 Conclusion **34**

List of Figures

2.1 noise transmission channel 2
2.2 eavesdropper model 3
2.3 Mutual Information of two correlated sources 4
2.4 Tanner Graph 5
2.5 BP algorithm decoding 7
2.6 The LLR value between [-3.5 3.5] 9
2.7 The density of LLR when a=+1 10
2.8 The density of LLR when a=-1 10

3.1 Tanner graph of the specific case 12
3.2 Distribution of LLR at message node 13
3.3 Distribution of LLR at parity node 14
3.4 Distribution of X1 at message node 15
3.5 Distribution of X1 at parity node 15
3.6 Distribution of X2 at message node 16
3.7 Distribution of X2 at parity node 16
3.8 Distribution of X3 at message node when X2 is positive 17
3.9 Distribution of X3 at message node when X2 is negative 18
3.10 Distribution of X3 at parity node when X2 is positive 18
3.11 Distribution of X3 at parity node when X2 is negative 19
3.12 Distribution of X4 at message node when X3 is positive 19
3.13 Distribution of X4 at message node when X3 is negative 20
3.14 Distribution of X4 at parity node when X3 is positive 20
3.15 Distribution of X4 at parity node when X3 is negative 21
3.16 Distribution of mid point at message node when X4 is positive 22
3.17 Distribution of mid point at message node when X4 is negative 22
3.18 Distribution of mid point at parity node when X4 is positive 23
3.19 Distribution of mid point at parity node when X4 is negative 23
3.20 Distribution of X5 at message node when X4 is positive 24
3.21 Distribution of X5 at message node when X4 is negative 24
3.22 Distribution of X5 at parity node when X4 is positive 25
3.23 Distribution of X5 at parity node when X4 is negative 25
3.24 2D distribution at message node 26
3.25 2D distribution at parity node 27
3.26 convolution when both distribution at $\phi = 0$ 27
3.27 convolution when both distribution at $\phi = \pi$ 28

3.28 2D convolution of one message node and one parity node 28
3.29 convolution when both distribution at $\phi = 0$ 29
3.30 convolution when both distribution at $\phi = \pi$ 29
3.31 2D convolution of one message node and one parity node 30
3.32 convolution when both distribution at $\phi = 0$ 30
3.33 convolution when both distribution at $\phi = \pi$ 31
3.34 2D convolution of one message node and one parity node 31
3.35 The Distribution after exponential 32
3.36 The Distribution after inverse hyperbolic tangent 33

Chapter 1

Introduction

Along with the increasing reliance upon the wireless transmission of private data, many institutes seek for a better way to deal with the transmission security since the data has no sufficient protection and is vulnerable to eavesdropping attacks from unauthorized or even adversarial users. Physical layer security comes into consideration, since Shannon's cipher model can be viewed as an upper-layer abstraction which assumes that the correction of transmission errors at the physical layer has already been dealt with. The pivotal idea behind physical layer security models is to revisit this assumption and to explicitly consider the presence of imperfect communication channels [2]. In other words, we say that the physical layer security explores the characteristics of the wireless channel to improve the transmission security.

Approaching the improvement by information theory, key generation and key reconciliation are introduced. In order to do the reconciliation procedure, we employ the Slepian-Wolf Coding and LDPC code. LDPC codes are capacity-approaching codes so that they allow the noise threshold to be set very close to the Shannon limit. The BP algorithm is applied to decode the LDPC code and the density evolution is used to deal with the density of the LLR. The general formulas in density evolution, nevertheless, only work under the consistent Gaussian situation. What Bob receives could not be guaranteed to be Gaussian, while in our case the distribution is obviously not consistent.

In order to obtain a solid base for the further research, this paper will discuss and simulate what will happen if the initial case is not consistent. We firstly analyze at the variable node side and then go to the check node side. The LLR forwarding to the check node comes from two parts, message nodes and parity nodes when the incoming channel observation distribution at parity node could be proven to be Gaussian. We hope to see that the final density distribution will tend to be Gaussian after several iterations.

Chapter 2

Background

2.1 Shannon theory and Perfect Secrecy

The year 1948 marks the birth of information theory. In that year, *Claude E. Shannon* published his revolutionary paper on the limits of reliable transmission of data over unreliable channels [1]. This paper formalized the concept of information, and established bounds for the maximum amount of information that can be transmitted over unreliable channels.

Given a communication channel, Shannon proved that there exists a value, called the capacity of the channel, such that reliable transmission is possible for rates arbitrarily close to the capacity, and reliable transmission is not possible for rates above the capacity.

Definition 1 (Channel capacity)**.** Channel capacity is the tight upper bound on the rate at which information can be reliably transmitted over a communication channel.

It is possible to transmit information over this channel reliably (with probability of error $\rightarrow 0$) if and only if

$$R = \frac{\#information\,bits}{channel\,use} < C$$

Let us recall a simple noisy transmission channel.

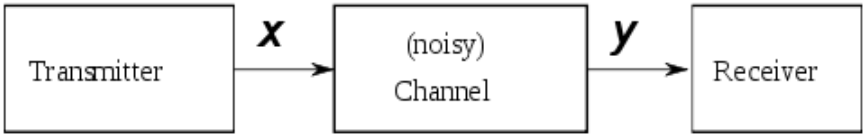

Figure 2.1: noise transmission channel

Let X and Y be the random variables representing the input and output of the channel, respectively. Let $P_{X,Y}(x, y) = P(y|x)P_X(x)$, which induces a mutual information $I(X;Y)$. The channel capacity can be expressed as

$$C = \sup_{P_X(x)} I(X;Y)$$

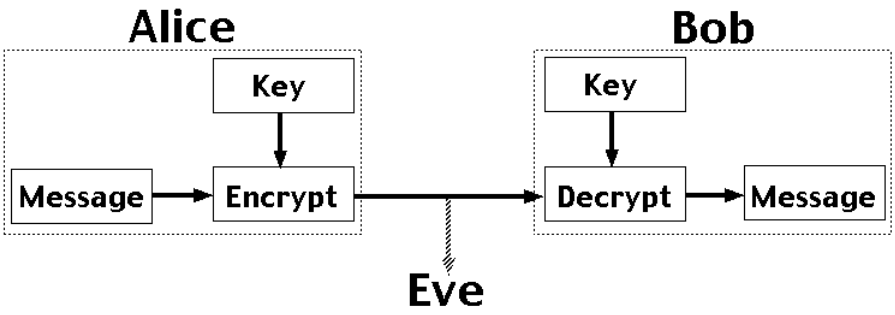

Figure 2.2: eavesdropper model

where the supreme is taken over all possible choices of $P_X(x)$.

Shannon later further studied confidentiality in information theory using the model of a cipher system including a transmitter, a legitimate receiver and an eavesdropper.

In the setting of Fig. 2.2, message is transmitted from Alice to Bob, the legitimate receiver, while keeping the message from the Eve, the eavesdropper. In order to ensure the secrecy of the message, Alice and Bob must share a key unknown to Eve, which can be used to encode the message into a codeword and decode the codeword back to the message. Respectively, the codeword, the shared key and the message are denoted by C,K,M.

If the message M is statistically independent of the codeword C intercepted by Eve, it could be called perfect secrecy, which means the mutual information $I(M;C)$ should be zero. Unfortunately, this situation only could be achieved if the entropy of secret key exceeds the entropy of the message. That is to say, in the transmission, the length of the key should be as long as, or even longer than the length of message, which largely limit the efficiency of the transmission. Because the upper layer structure, Shannon Model, cannot provide a solution to the inefficiency problem, physical layer security is revisited.

There are normally several different approaches used for the design and analysis of physical layer security techniques [2]:
1. Information Theoretic Approaches: introduces capacity-achieving methods and coding schemes for secure communication, as well as secret key generation and agreement over wireless channels

2. Signal Processing Approaches: covers recent progress in applying signal processing techniques to design physical layer security enhancements

3. Game Theoretic Approaches: discusses the applications of game theory to analyze and design wireless networks with physical layer security considerations

4. Graph Theoretic Approaches: presents the use of tools from graph theory and stochastic geometry to analyze and design large-scale wireless networks with physical layer security constraints

In our case, we choose the key generation and reconciliation for further research.

2.2 Key generation and reconciliation by Slepian-Wolf coding

Key generation is the process of generating keys for cryptography. A key is used to encrypt and decrypt the data. In order to maintain the secrecy of the information, key should change every time. While it is impossible to change the channel, one way to vary the key every time is to use RECAP antenna arrays. In [3], the method for generating the key through the RECAP antenna structures was described. Though Alice and Bob simultaneously generate the key, due to Channel State Information (CSI) differences, Bob would still quantize erroneous results compared with Alice's, assuming Alice's key to be the correct one.

In order to correct key differences, key reconciliation procedures are required on Bob's side. Since the key from Alice's side is assumed to be correct, Alice could send parity bits to Bob over the channel so that Bob could correct his erroneous measurement to the correct one according to parity bits. For ensuring the length of parity bits and the key reconciliation procedures, the Slepian-Wolf coding based on Low-Density Parity-Check (LDPC) code is applied.

Slepian-Wolf coding is a type of source coding with side information, which can be seen as a key reconciliation option when one legitimate user (Alice) compresses her key information x and sends side information to the other legitimate user (Bob) such that he can decode the correlated information y using the received side information. The total number of bits (M) that have to be sent for key reconciliation is given by the Slepian-Wolf lower bound [4].

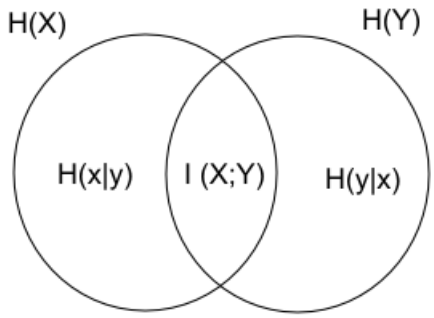

Figure 2.3: Mutual Information of two correlated sources

$$M = H(x \mid y)bit \tag{2.1}$$

For an n bit key, the number of reconciliation bits M_p are

$$M_p \geq nM = nH(x \mid y)[bit] \tag{2.2}$$

Therefore, the conditional entropy between the channel estimates of Alice and Bob represents an indicator for the number of parity bits needed.

2.3 LDPC codes

For any code $C(n, k)$, it is a linear code if there is a linear relationship between information symbol and the redundancy, that is to say it could be described by a linear function. The Low-Density Parity-Check code is a type of linear code. Using the generator matrix G, it could generate the transmitted information sequence, the codeword C. Corresponding to the generator matrix G, there is a parity-check matrix H, which needs to be fulfill the equation $Hc^t = 0$. In other word, all codewords construct the null space of parity-check matrix H.

The LDPC code is constructed using a sparse bipartite graph. The number of non-zero elements in the parity-check matrix H is small, that is the reason why LDPC code named low-density parity-check code. Due to the sparsity of the parity-check matrix H, the tanner graphs of each LDPC codes have different Girth which largely effect the decoding performance by, e.g. Belief Propagation algorithm, though we do not discuss here.

The Tanner graph mentioned above is widely used to represent the parity-check matrix H in the LDPC code. It is a bipartite graph with two sets of nodes which state constraint with the specific error correcting codes. The Tanner graph of the Hamming code with the check matrix:

$$H = \begin{array}{c} \\ c1 \\ c2 \\ c3 \\ c4 \end{array} \begin{array}{c} \begin{array}{ccccccc} v1 & v2 & v3 & v4 & v5 & v6 & v7 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

is shown in Fig. 2.4

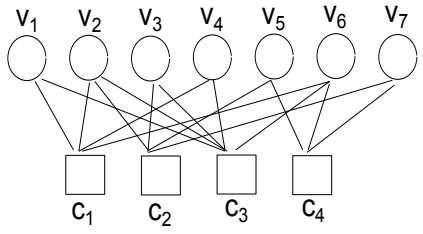

Figure 2.4: Tanner Graph

The Tanner Graph in the Fig. 2.4 could represent the corresponding H matrix: the circles denote the variable nodes and the boxes denote the check nodes. An LDPC code could be defined as regular one if it could be denoted by three parameters (N, t_c, t_r), in other words, the N bits codeword has a matrix H with exactly t_c and t_r ones per column or row. Correspondingly, the irregular LDPC code does not fulfill the above situation and represented by polynomials.

The degree distribution representation for variable-nodes is given by

$$\lambda(x) = \sum_{i=2}^{t_{cmax}} \lambda_i x^{i-1} \ . \tag{2.3}$$

The degree distribution representation for check-nodes is

$$\rho(x) = \sum_{j=2}^{t_{rmax}} \rho_j x^{j-1} \ , \tag{2.4}$$

where λ_i and ρ_j are the proportions of edges of the graph connected to bit nodes and check nodes of degree i and j, respectively, and t_{cmax} and t_{rmax} are the maximum numbers of edges linked to bit and check nodes.

2.4 Belief-Propagation Decoding

While the Maximum likelihood decoding is possible for LDPC code, the complexity does not allow for efficient implementation. Then we choose Belief-Propagation (BP) decoding as the sub-optimum choice.

BP algorithm is an iterative algorithm which spreads along edges forwarding probability or logarithmic likelihood ratios (LLR). The LLR shows the estimation of the bit-value of the associated variable node to which it belongs [5].

Definition 2 (logarithm likelihood ratio)**.** The messages over edges are one dimensional and called LLR, for log-likelihood ratio. The LLR of a message coming from a variable node will be denoted by v, and u will denote a message coming from a variable node. They are defined by

$$v = \log \frac{p(y|a=0)}{p(y|a=1)} \tag{2.5}$$

$$u = \log \frac{p(y'|a'=0)}{p(y'|a'=1)} \tag{2.6}$$

where a is the bit value of the node and y denotes all the information available to the node up to the present iteration obtained from edges other than the one carrying v. a' is the bit value of the variable node that gets the message from the check node up to the present iteration obtained from edges other than the one carrying [6].

Hence, if the LLR greater than zero, the corresponding estimate of a is zero; otherwise, the estimate should be one. The amplitude of the LLR demonstrates the significance of the estimate. The larger LLR value is, the higher possibility for Bob estimate the correct information.

In the BP algorithm, each decoding iteration consists of two stages. Starting with the variable node, the first stage processes the information from the previous iteration and transmit the new information to the check nodes. The second stage processes on check nodes and sends it back to the variable nodes.

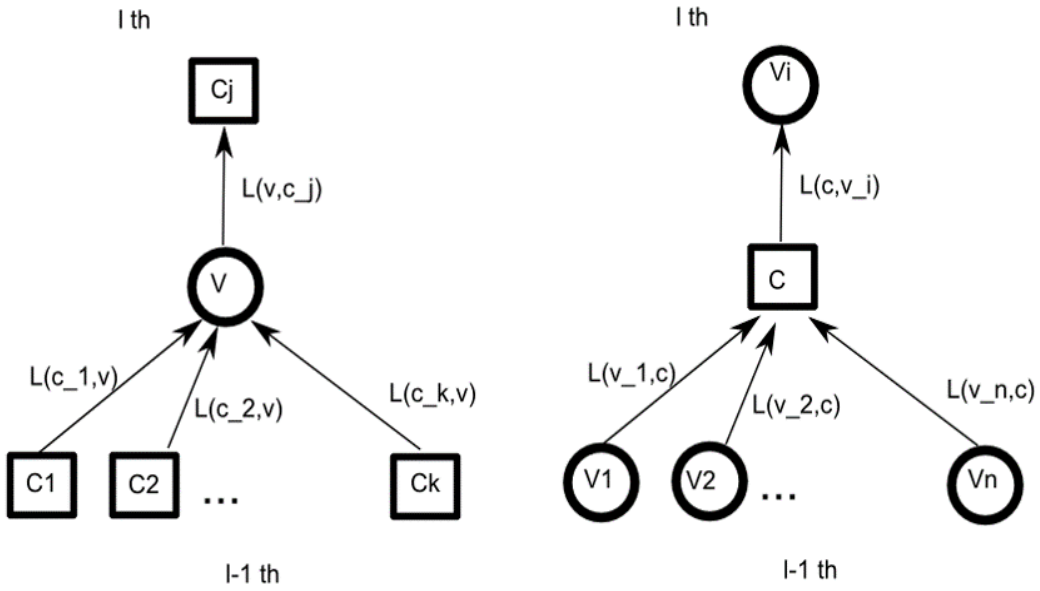

Figure 2.5: BP algorithm decoding

The left plot in Fig. 2.5 represents a decoding iteration at a variable node:

$$L_{v_i c_j}^{(l)} = L_{ch} + \sum_{k \neq j} L_{c_k v_i}^{(l-1)} \tag{2.7}$$

where,

$L_{v_i c_j}$: the LLR (over the l^{th} iteration coming) out of a bit node
L_{ch} : the LLR of the channel observation
$L_{c_k v_i}$: the LLR coming out of a check node

The right plot in Fig. 2.5 represents a decoding iteration at a check node:

$$L_{c_i v_j}^{(l)} = 2 \tanh^{-1} \left(\prod_{k \neq j} \tanh \left(\frac{L_{v_k c_j}^{(l)}}{2} \right) \right) , \tag{2.8}$$

where,

$L_{c_i v_j}$: the LLR (over the l^{th} iteration coming) out of a check node
$L_{v_k c_j}$: the LLR coming out of a bit node

2.5 Density Evolution

Density Evolution is an optimal method to derive the degree distribution. Normally based on the assumption of Gaussian input distributions, density evolution has been widely used to analyze and design LDPC codes with good performance for classical point-to-point communication applications, e.g., communications.over AWGN channels or binary erasure channels (BECs) [7] [8]. It provides the mutual information between iterations and could be used to estimate the quality of decoder.

If the density is Gaussian distribution, a simple expression could be used to describe the mutual information by $J(m)$:

$$J(m) = 1 - E_v(\log_2(1 + e^{-v})) = x_v, \quad v \sim N(m, 2m) \; . \tag{2.9}$$

$J(m)$ is a continuous and strictly monotonous function, so J^{-1} exists and allows for computing the mean m of the LLR from the mutual information x_v [6]. Because the consistency can be preserved during the iteration, BP decoding on check node and variable nodes can be directly use the J function:

$$L_{cv}^{(l)} = \sum_{j=2}^{d_{\max_j}} \rho_j J\left((j-1)\, J^{-1}\left(1 - L_{vc}^{(l-1)}\right)\right) \tag{2.10}$$

$$L_{vc}^{(l)} = \sum_{i=2}^{d_{\max_j}} \lambda_i J\left(\frac{2}{\sigma^2} + (i-1)\, J^{-1}\left(L_{cv}^{(l-1)}\right)\right) \tag{2.11}$$

ρ : check node degree distribution polynomial
λ : variable node degree distribution polynomial
$J(\cdot)$: $J(m) = 1 - E_v(\log_2(1 + e^{-v})) = x_v, \quad v \sim N(m, 2m)$;
$\frac{2}{\sigma^2}$: mean m of consistent Gaussian message $L_{channel,intrinsic} \sim L_{ch,i}$.

Nevertheless, if the density is not consistent, the mutual information function should be represented in another way:

$$x_v = \int_R f(v|a=+1)\, \log_2\left(\frac{2f(v|a=+1)}{f(v|a=+1) + f(v|a=-1)}\right) \mathrm{d}v \; . \tag{2.12}$$

where,

v : the message measurement

x_v : the mutual information between v and input a

In order to check whether the measurement is consistent Gaussian, he plots the measurement value in Fig. 2.6:

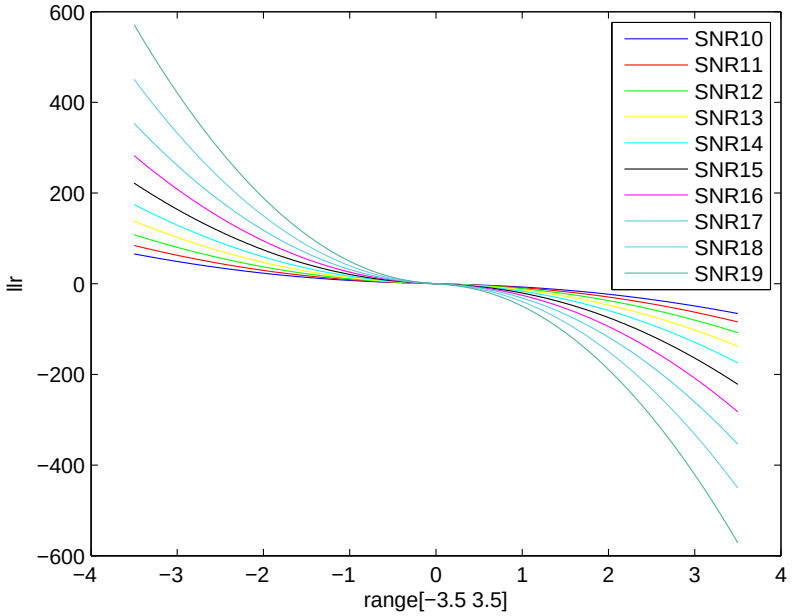

Figure 2.6: The LLR value between [-3.5 3.5]

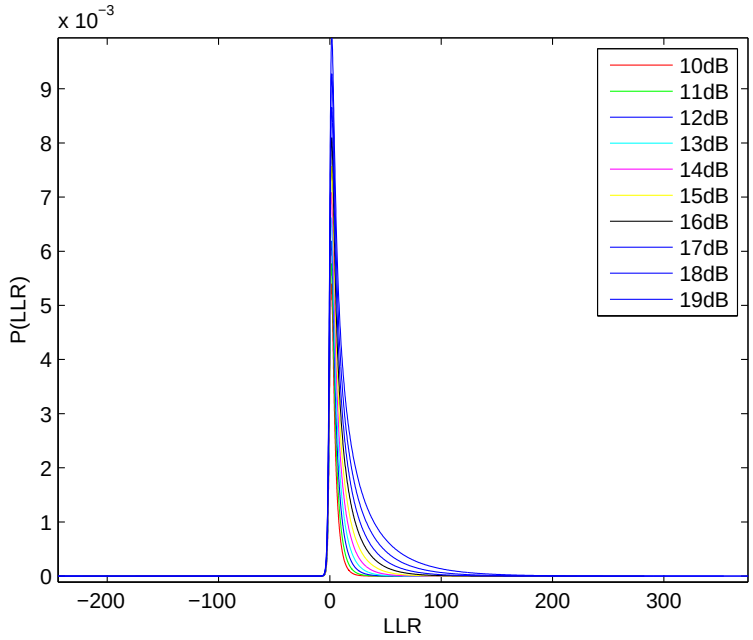

Figure 2.7: The density of LLR when a=+1

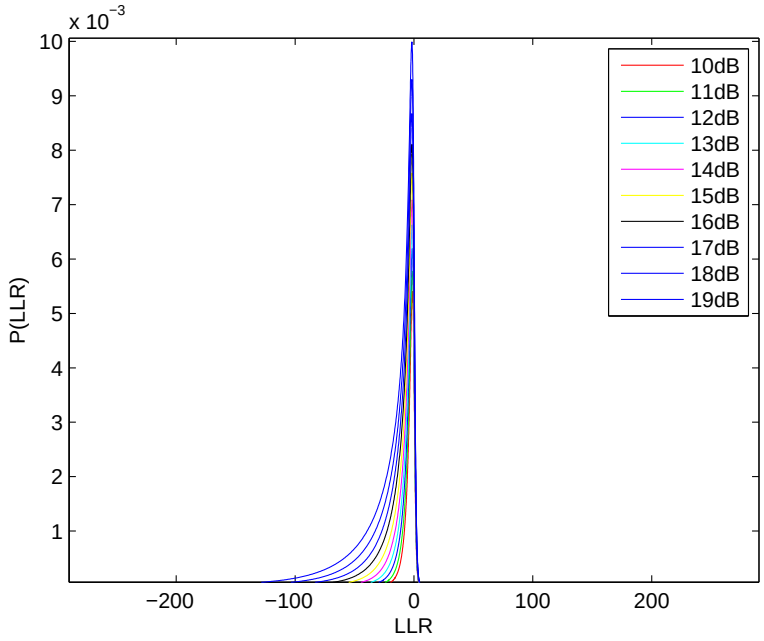

Figure 2.8: The density of LLR when a=-1

The Fig. 2.7 is the probability when a=+1, while the Fig. 2.8 is when a=-1. However, no matter under which situation, the distribution is non-Gaussian. Therefore, the eqn.([2.10]) and eqn.(2.11) cannot be directly applied and every step need to go through the eqn.(2.12]) individually.

Chapter 3

Calculation and Simulation

3.1 Decoding at a Variable Node

Recall the BP decoding algorithm at the variable node side, the eqn.(2.7) combines the LLR from channel observation with the LLR coming from the check node in the $(l-1)th$ iteration. Due to the operation on L_{vc} is summation and the corresponding transform operation in density domain of summation is convolution. It is not so hard to see the contributing PDFs:

$$L^{(l)}_{v_i c_j} : \ f(L^{l-1}_{cv}) \star f(L_{ch,i}) \ = \ f(L^{l}_{v_i c_j}) \ . \tag{3.1}$$

LLRs are quantities which concern about both the binary source distributions $a_x = +1$ and $a_x = -1$ together for the received value and reflect the estimation for the transmitted bit in the sign of the LLR [9]. Therefore, the eqn.(2.8) and eqn.(2.7) are sufficient to demonstrate the estimate value. Nevertheless, the densities when $a_x = +1$ and $a_x = -1$ still need to be calculated separately. We define:

$$f(L^{(l)}_{v_i c_j} | a_x = +1) \ = \ f(L^{l-1}_{cv} | a_x = +1) \star f(L_{ch,i} | a_x = +1) \ , \tag{3.2}$$

$$f(L^{(l)}_{v_i c_j} | a_x = -1) \ = \ f(L^{l-1}_{cv} | a_x = -1) \star f(L_{ch,i} | a_x = -1) \ . \tag{3.3}$$

The above equations could describe the density evolution at the variable node side.

3.2 Decoding at a Check Node

Now, recall the BP decoding algorithm at the check node side, the eqn.(2.8), however, is much more complex than eqn.(2.7). Because there are non-linear operations, e.g. inverse hyperbolic tangent function, in the formula. We reform the equation for a better understanding.

The first difficulty of the expression is the multiplication operation. While the addition is analogous to the convolution in density domain, the corresponding operation of multiplication in density domain, the joint distribution, needs more concern due to a fraction inside the integral equation.

Here is the definition of joint distribution. if X and Y are two independent, continuous random variables, described by probability density function f_X and f_Y then the joint distribution of $Z = XY$ is

$$f_Z(z) = \int_{-\infty}^{\infty} f_X(x) f_Y(z/x) \frac{1}{\mid x \mid} dx$$

As we known, if a denominator in a fraction is zero, then the fraction is undefined at that point. However, in our case, the denominator is the value of LLR, which could certainly be zero. Therefore, a multiplication should bed replaced by the summation. Due to the characteristic of logarithm, the logarithm of a product is the sum of the logarithm of the factors:

$$\log_b(xy) = \log_b(x) + \log_b(y)$$

When the logarithm is applied, the exponential should also applied so that the equation will not change. We could convert the original equation by $\exp(\log(mutiplication\ of\ equation)) = \exp(summation(\log(equation))$. Then the original expression turns to be

$$L_{c_i v_j}^{(l)} = 2 \tanh^{-1} \exp\left(\sum_{k \neq j} \log\left(\tanh\left(\frac{L_{v_k c_j}^{(l)}}{2} \right) \right) \right) \tag{3.4}$$

After converting the equation to the logarithm-exponential form, the second problem occurs due to the logarithm. In real domain, the logarithm operation is only valid when the variable larger than zero. However, LLR estimates the channel according to the sign. Hence, there is possibility for LLR being negative. In order to solve it, one way is to work in the complex domain and to deal with the real part and imaginary part separately:

$$L_{c_i v_j}^{(l)} = 2 \tanh^{-1} \exp\left(\sum_{k \neq j} \log\left(\mid \tanh\left(\frac{L_{v_k c_j}^{(l)}}{2} \right) \mid \right) + j\phi_k \right) \tag{3.5}$$

The eqn.(3.5) is what should be dealt with. And in order to simplify the calculation, we provide a specific situation with tanner graph:

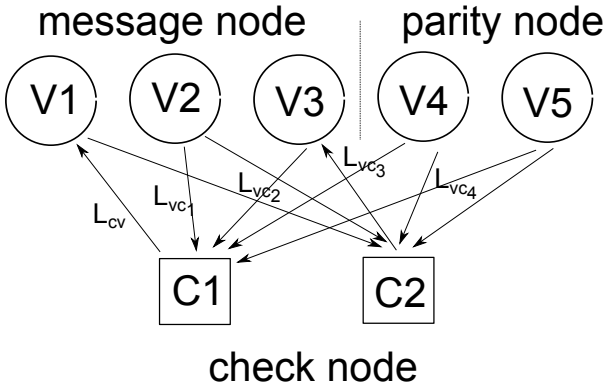

Figure 3.1: Tanner graph of the specific case

We select check node $C1$ as our case. Assume the following condition:
1. the bit value of the node $a = +1$

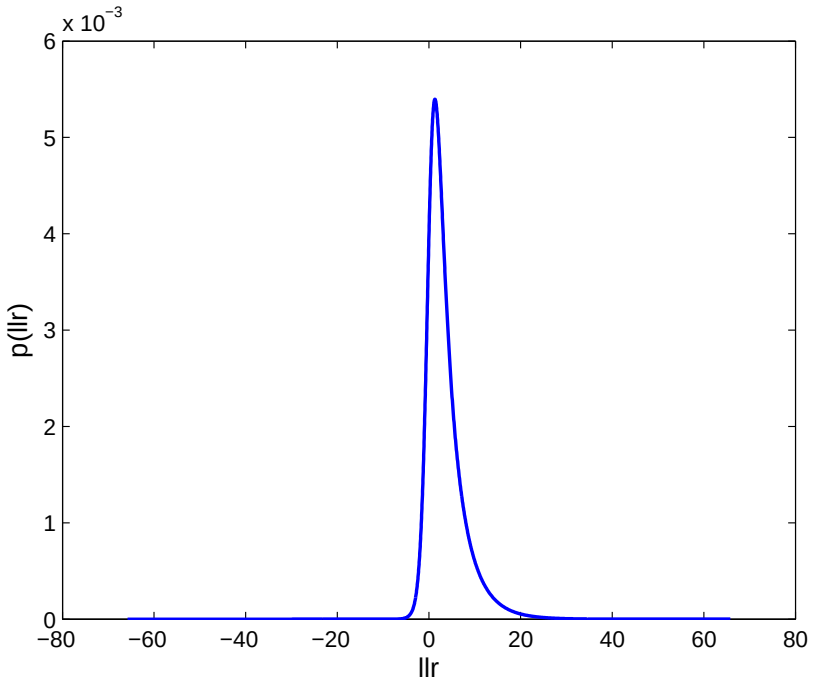

Figure 3.2: Distribution of LLR at message node

2. SNR value=10dB

The density transformation which is analogous to the random variable operation can be achieved via dividing the slope or changing the interval. When we attempted to apply the former one, we noticed that the slope after the operation of hyperbolic tangent would go to infinity, which could not be divided. Therefore, We choose the latter one. We define the $f(y)$ and $f(x)$ as below :

$$f(y) := f(L_{cv}|a = +1);$$

$$f(x_i) := f(L_{vc_i}|a = +1)$$

Note that the data is discrete while the transform operations deal with the intervals. Each point measured by Bob is concerned as the bound of every interval.

3.3 Calculation and Simulation

3.3.1 General Operation

We split the eqn.(3.5) into several basic step and the original data are shown as Fig. 3.2 and Fig. 3.3

The distribution at the message is non-Gaussian and not consistent while the distribution at parity node is Gaussian. Assume the information in parity node side comes from an AWGN channel with $y = a + n$,
a : the discrete input, n: the noise, y: output The Gaussian density

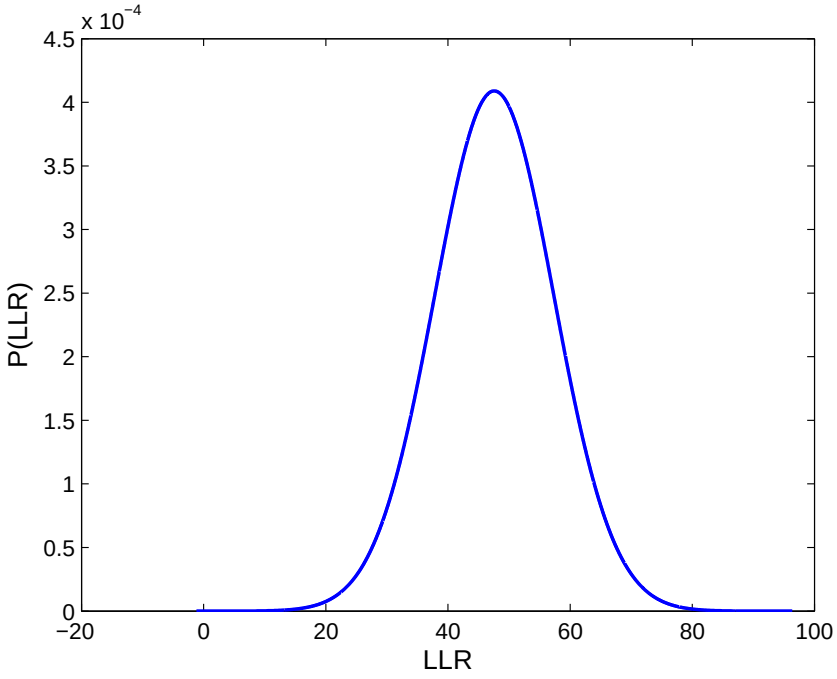

Figure 3.3: Distribution of LLR at parity node

$$p(y|A=a) = \frac{1}{\sqrt{2\pi}\rho_n} \exp(\frac{-(y-a)^2}{2\rho_n^2})$$

leads to the LLR

$$L(y|a) = \ln \frac{p(y|a=+1)}{p(y|a=-1)} = \frac{2}{\rho_n^2} * y = \frac{2}{\rho_n^2} * (a+n)$$

Hence

$$\mu_L = \frac{2}{\rho_n^2} \cdot a, \quad and \quad \rho_L^2 = (\frac{2}{\rho_n^2})^2 \cdot \rho_n^2 = \frac{4}{\rho_n^2}$$

This demonstrates that mean has relation with the variance

$$\mu_L = \frac{\rho_L^2}{2}$$

Therefore, we say the distribution in parity node is Gaussian [6].

Step 1: $x1_i = L_{vc\_i}/2$
Because we change the interval rather than dividing the slope, here the x shrink to the half. The operation is linear so we do not need consider other factors.

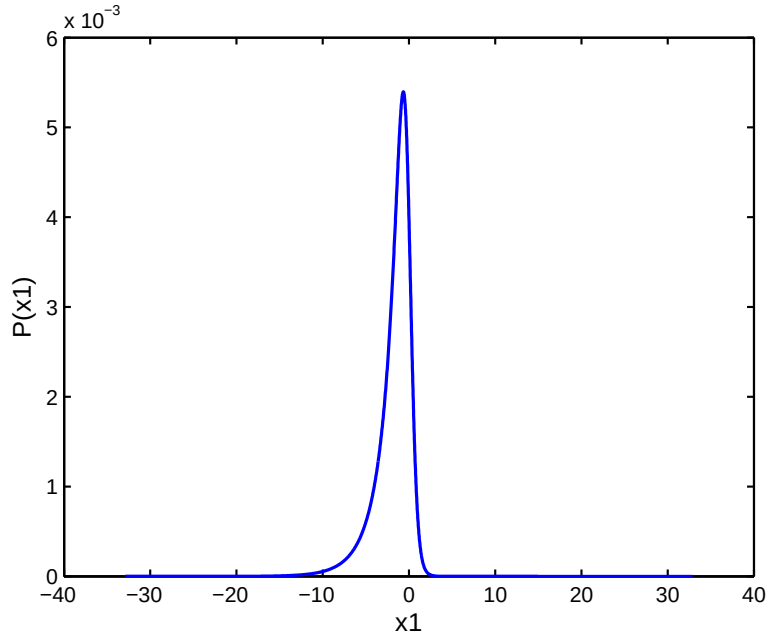

Figure 3.4: Distribution of X1 at message node

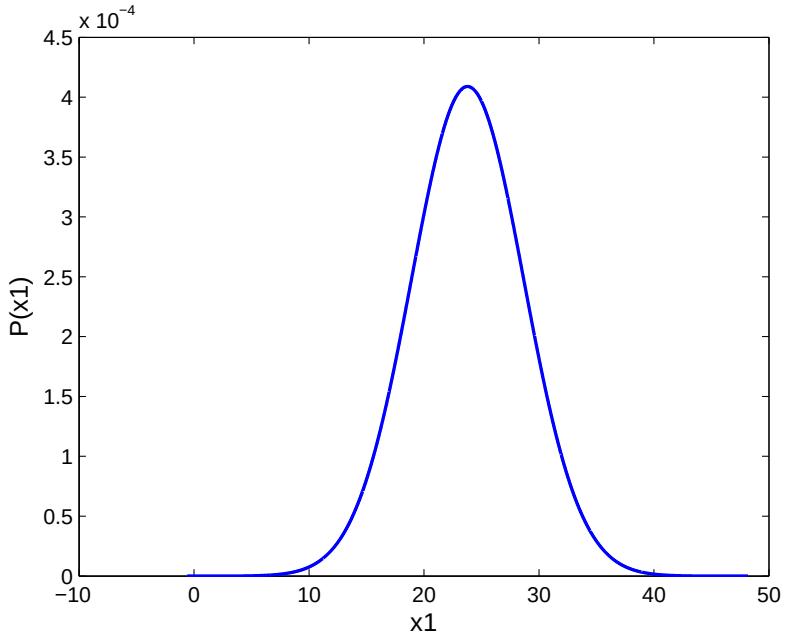

Figure 3.5: Distribution of X1 at parity node

Step 2: $x2_i = \tanh(x1_i)$
The hyperbolic tangent is a non-linear operation and due to its characteristic, all the interval value will be bounded between -1 and $+1$. Hence, when $x2 = \pm 1$, there are many corresponding y

value. Note that the plots only show the distribution of the LLR, not a function. So it still could provide the information we want.

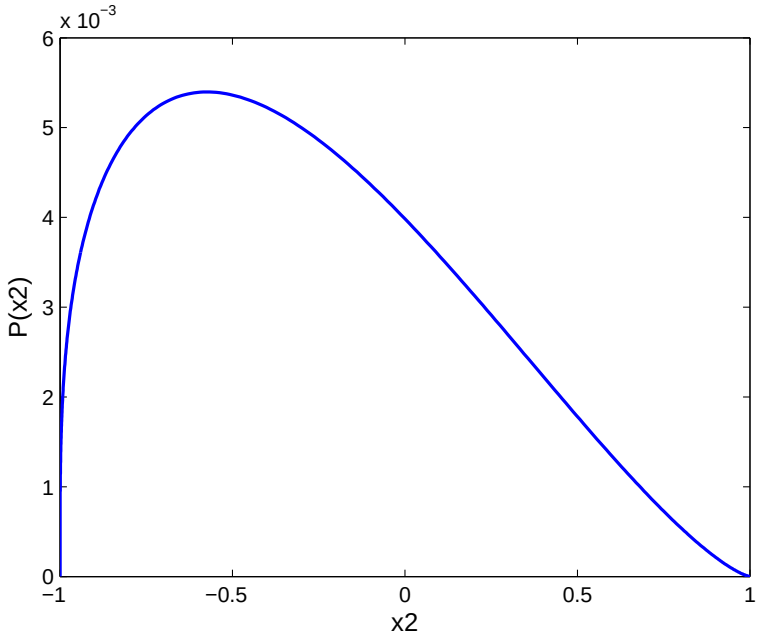

Figure 3.6: Distribution of X2 at message node

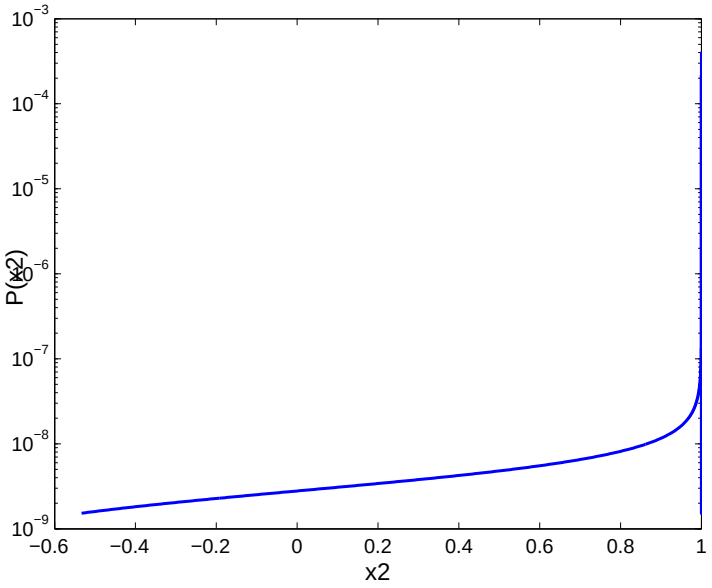

Figure 3.7: Distribution of X2 at parity node

3.3.2 Calculation in the Complex Domain

Step 3: $x3_i = |(x2_i)|$
From this step, all measurement will be treated separately. And from the eq[3.5], we see the value will be expressed as $R\angle\phi$. The ϕ of negative part of $x2$ is π while the positive is 0. Fig. 3.8 and Fig. 3.9 shows the plots at message node side and Fig. **??** and Fig. 3.11 shows the plots at parity node side:

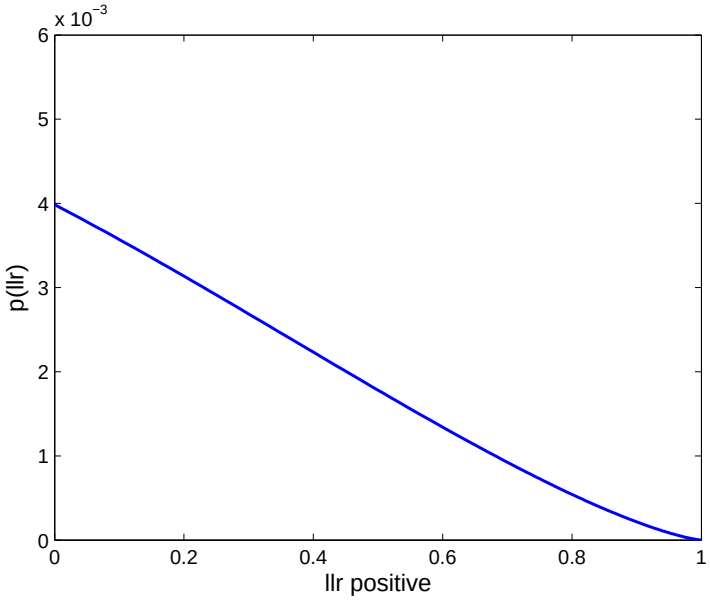

Figure 3.8: Distribution of X3 at message node when X2 is positive

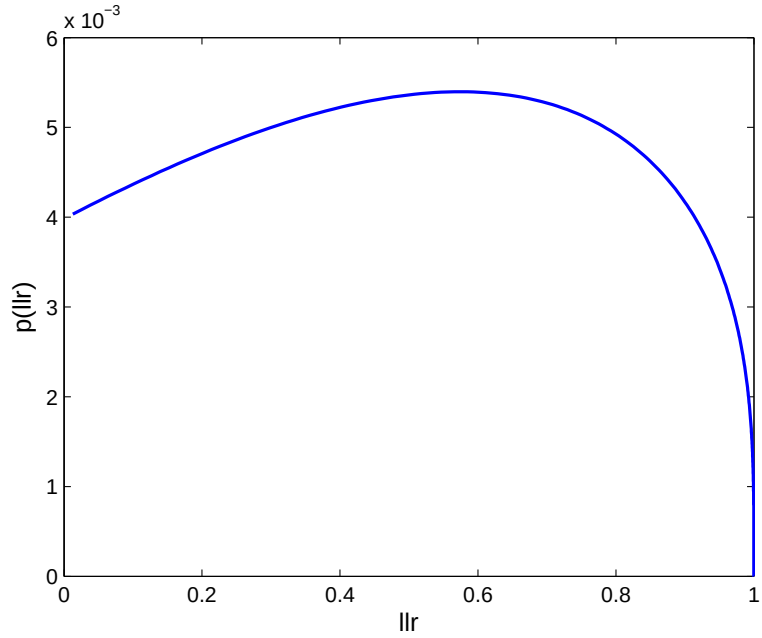

Figure 3.9: Distribution of X3 at message node when X2 is negative

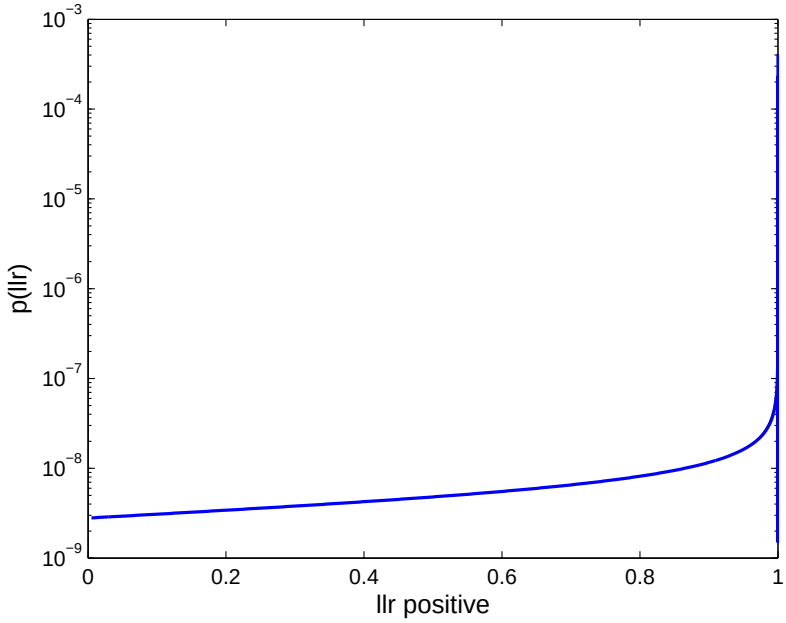

Figure 3.10: Distribution of X3 at parity node when X2 is positive

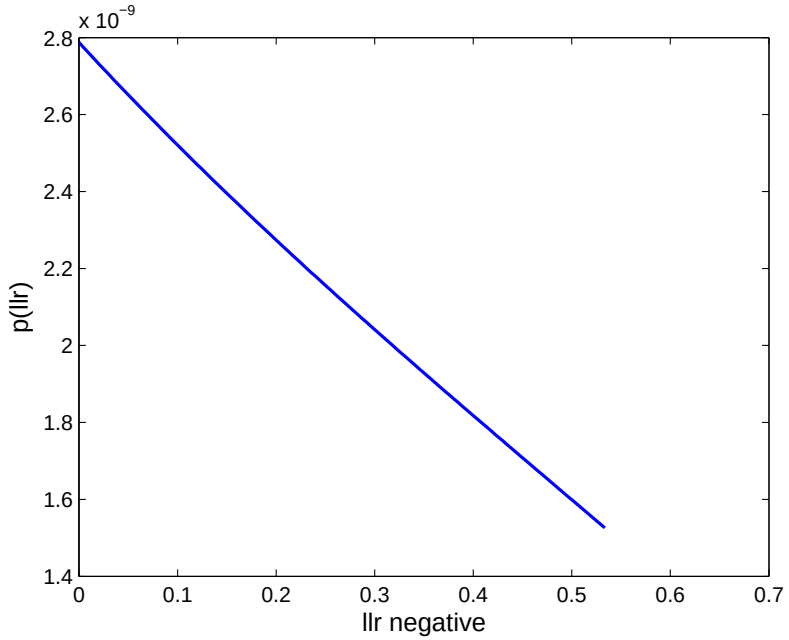

Figure 3.11: Distribution of X3 at parity node when X2 is negative

Step 4: $x4_i = \log(x3_i)$
In this step, we only need to apply the logarithm operation and see what will happen.

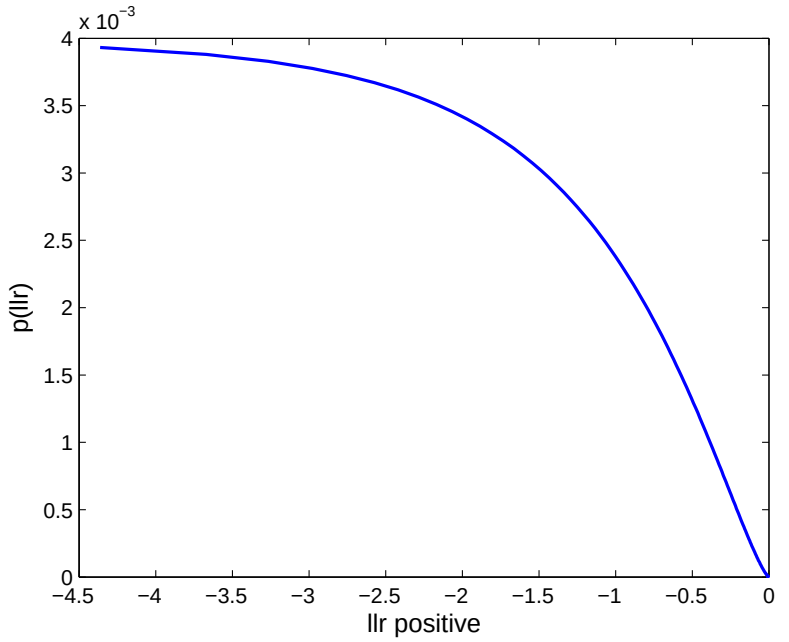

Figure 3.12: Distribution of X4 at message node when X3 is positive

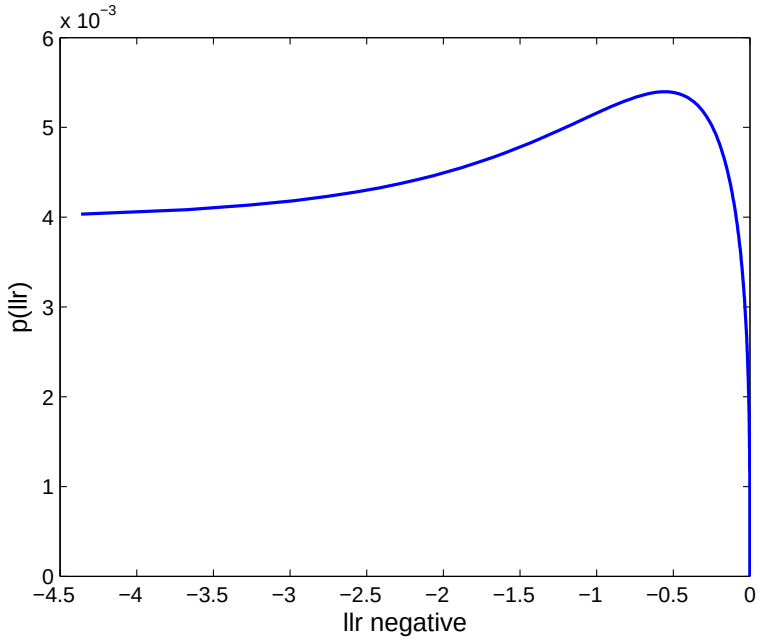

Figure 3.13: Distribution of X4 at message node when X3 is negative

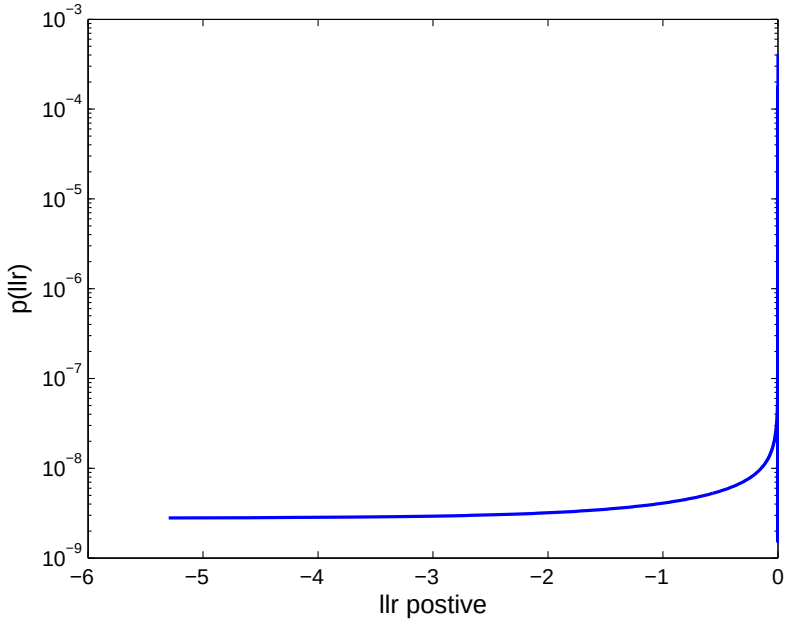

Figure 3.14: Distribution of X4 at parity node when X3 is positive

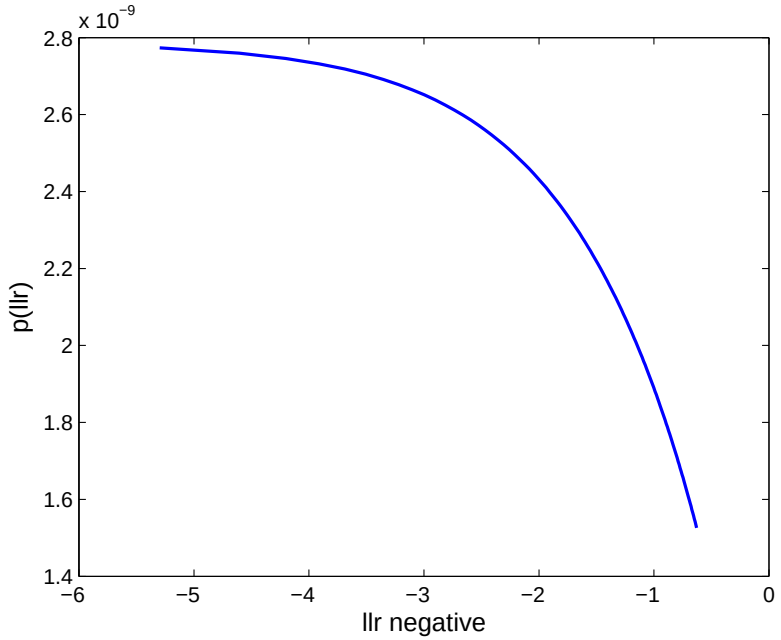

Figure 3.15: Distribution of X4 at parity node when X3 is negative

3.3.3 2D Convolution

Step 5: 2D convolution
But before the convolution, it is significant to rescale the x-axis into same vector space, otherwise it is impossible to do the convolution under discrete case. The mid-points represent the value in this interval so when rescaling, everything should be transform to mid-points then convert back to the bound (minimum and maximum value).

New scale: the range from -5 to 0 with the resolution 0.01

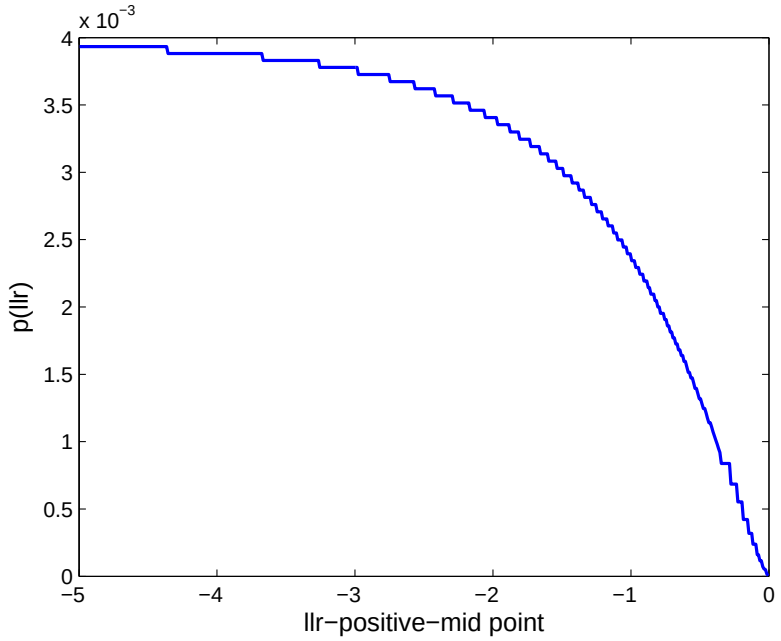

Figure 3.16: Distribution of mid point at message node when X4 is positive

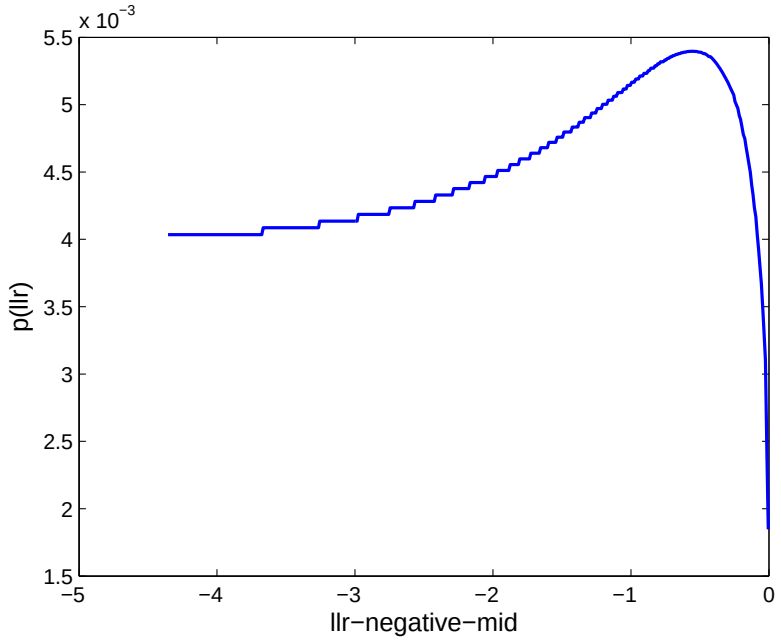

Figure 3.17: Distribution of mid point at message node when X4 is negative

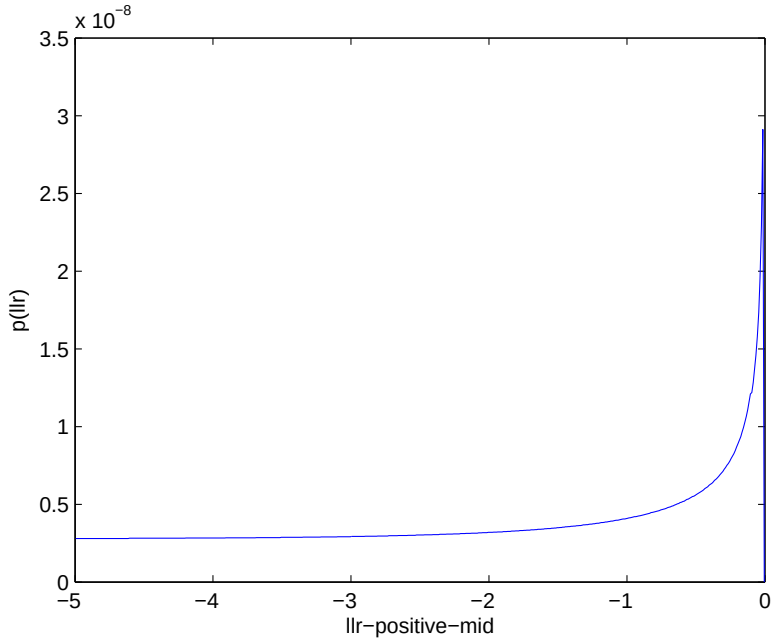

Figure 3.18: Distribution of mid point at parity node when X4 is positive

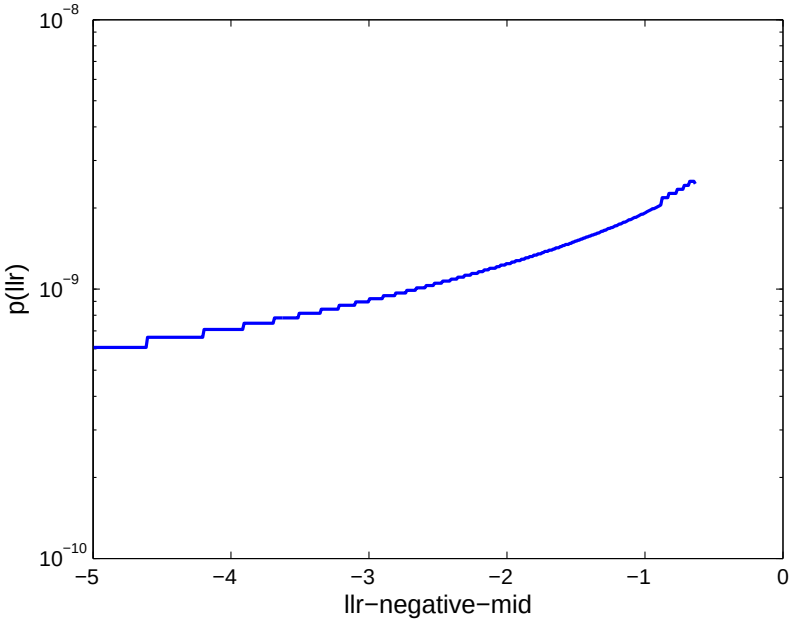

Figure 3.19: Distribution of mid point at parity node when X4 is negative

Then we convert the value in mid point to the bound value.

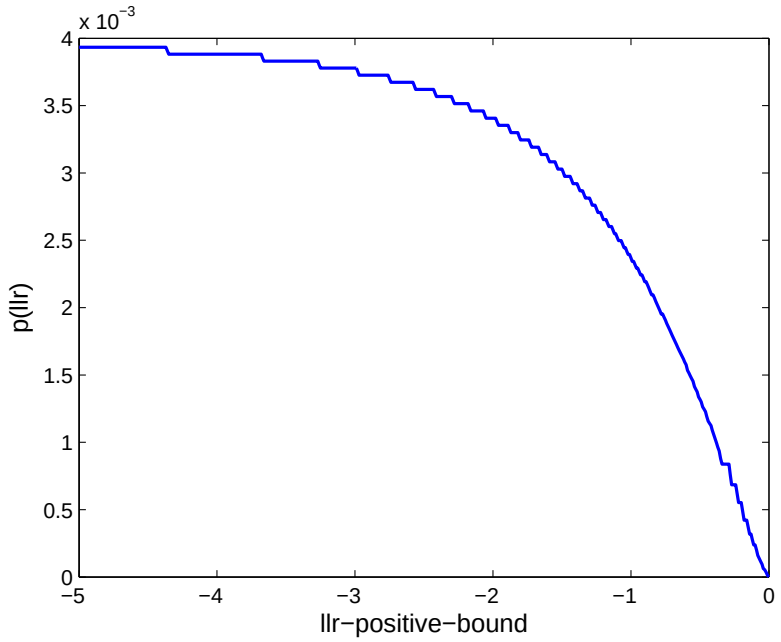

Figure 3.20: Distribution of X5 at message node when X4 is positive

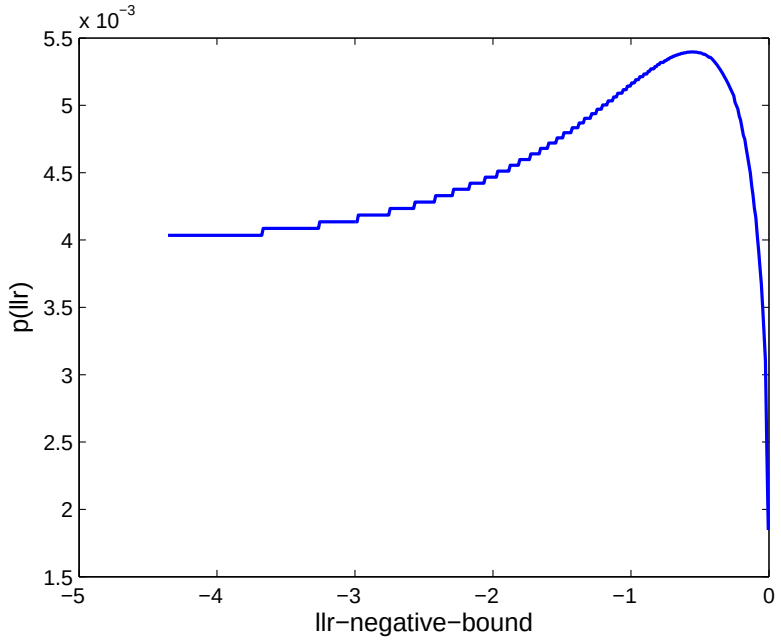

Figure 3.21: Distribution of X5 at message node when X4 is negative

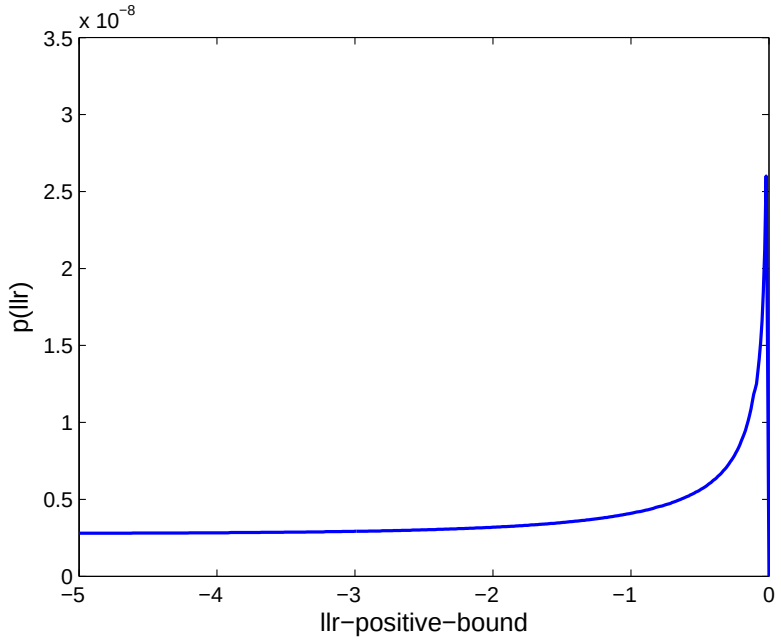

Figure 3.22: Distribution of X5 at parity node when X4 is positive

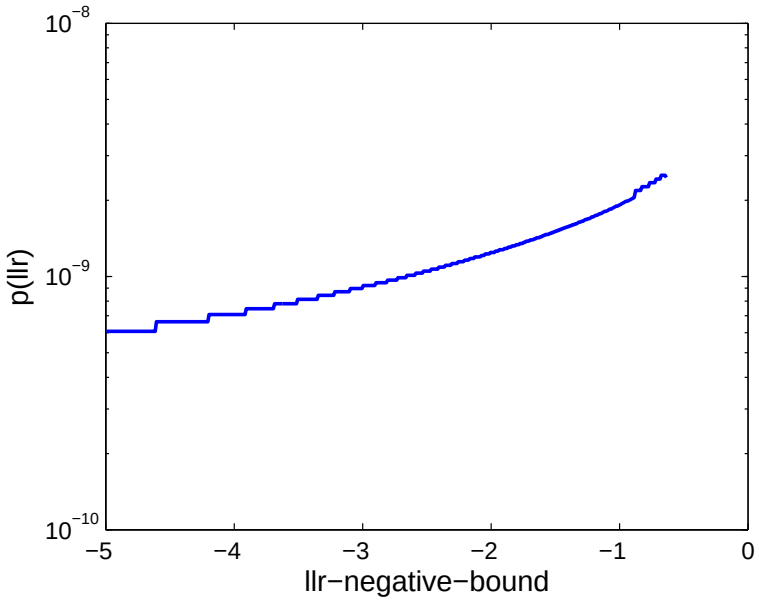

Figure 3.23: Distribution of X5 at parity node when X4 is negative

After the vectors are in the same space, the convolution should be performed based on those vectors. The individual densities can be written as (x stands for the real part, y for the imaginary, "|" means condition). Assume $f_1(x, y)$ and $f_2(x, y)$ represent any distribution in 2D convolution

$$f_1(x, y) = f_1(x|y = 0)\delta(y - 0) + f_1(x|y = \pi)\delta(y - \pi)$$

Let $f_2(x, y)$ be formulated likewise. Then, the convolution is

$$f(x', y') = \int_x \int_y f_1(x, y) \cdot f_2(x' - x, y' - y \text{ modulo } 2\pi)\ dydx$$

What will then be the individual results

$$\begin{aligned} f(x'|y' = 0) &= \int_x f_1(x|y = 0) \cdot f_2(x' - x|y = 0) + f_1(x|y = \pi) \cdot f_2(x' - x|y = \pi)\ dx \\ f(x'|y' = \pi) &= \int_x f_1(x|y = 0) \cdot f_2(x' - x|y = \pi) + f_1(x|y = \pi) \cdot f_2(x' - x|y = 0)\ dx \end{aligned}$$

The 2D distributions we have are:

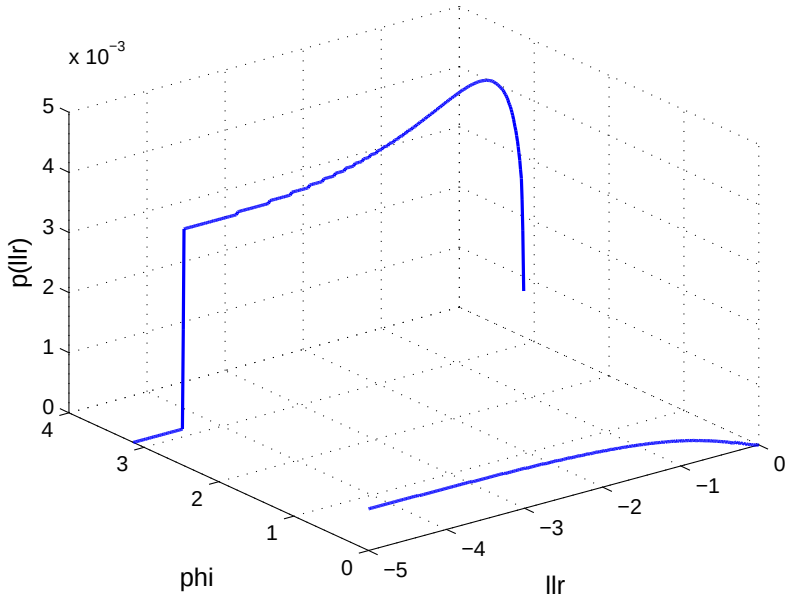

Figure 3.24: 2D distribution at message node

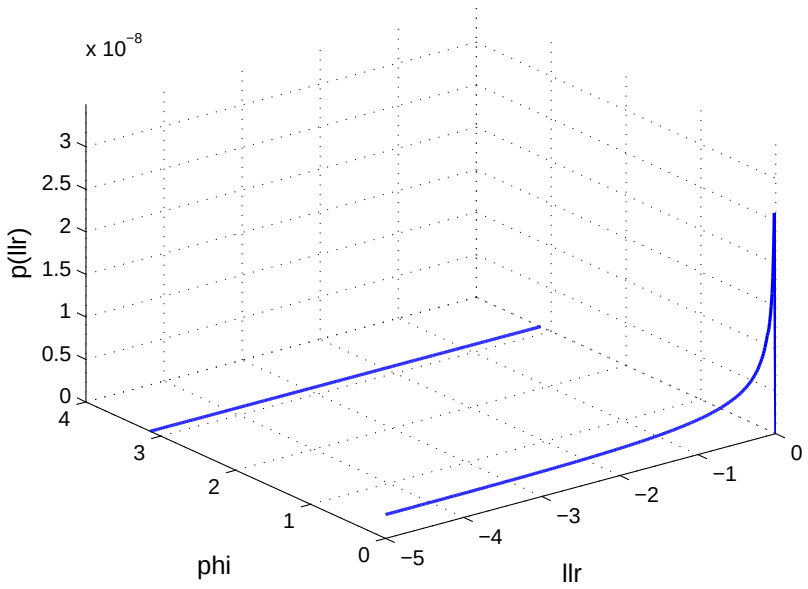

Figure 3.25: 2D distribution at parity node

According to the principle of 2D convolution we derived above, we could obtain the plots after the convolution of one message node and one parity node as below:

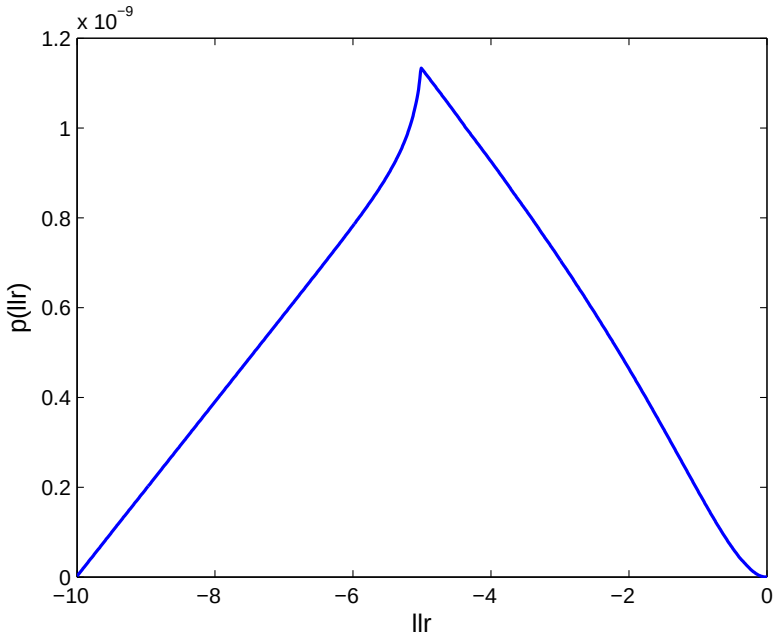

Figure 3.26: convolution when both distribution at $\phi = 0$

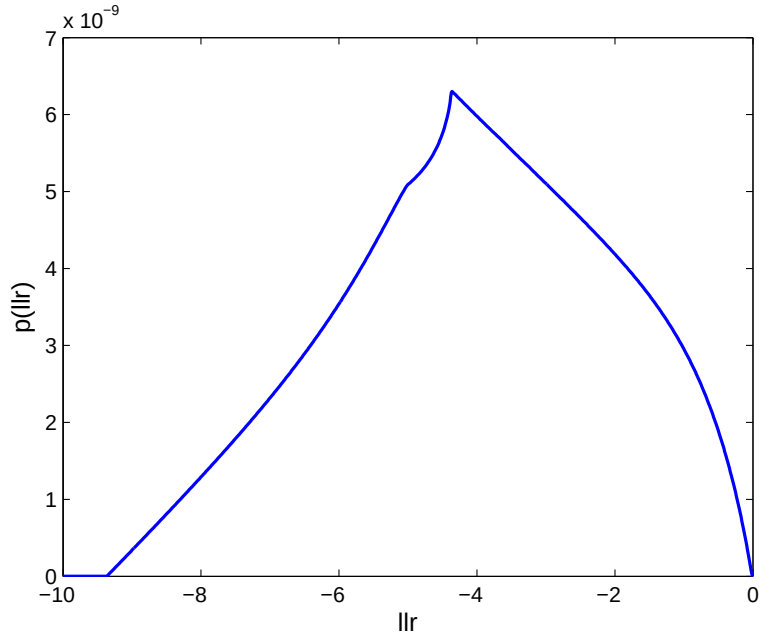

Figure 3.27: convolution when both distribution at $\phi = \pi$

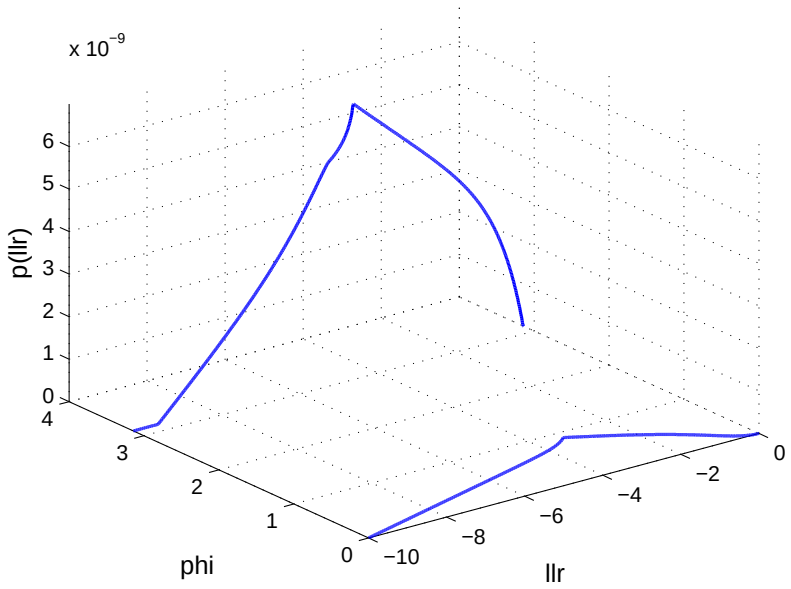

Figure 3.28: 2D convolution of one message node and one parity node

From Fig. 3.28, we could see the distributions are still not consistent and the shape is almost like a triangle. Then we convolve the former 2D distribution with one more parity node, which means the convolution includes distribution from two parity nodes and one message node.

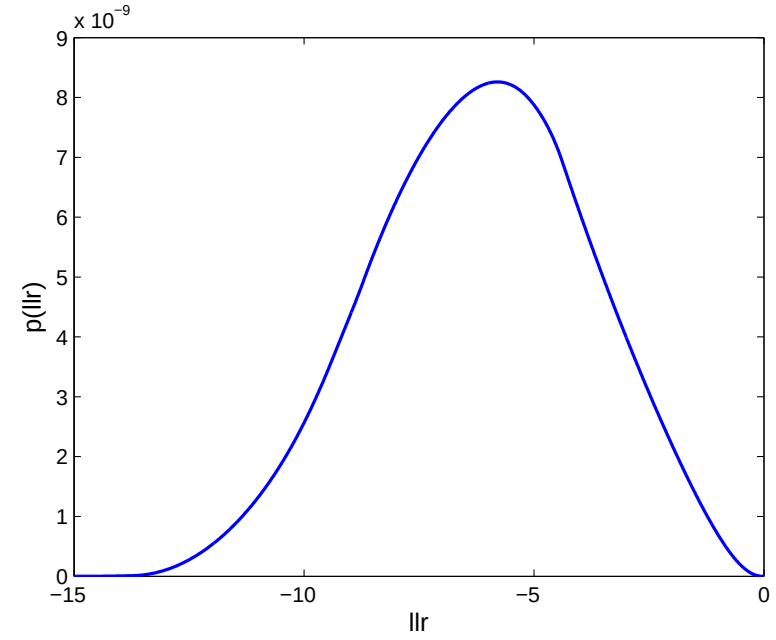

Figure 3.29: convolution when both distribution at $\phi = 0$

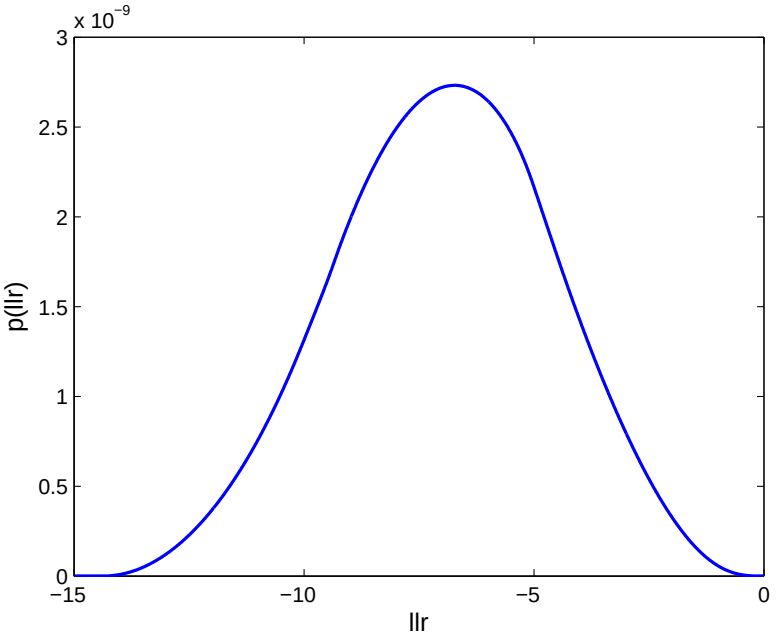

Figure 3.30: convolution when both distribution at $\phi = \pi$

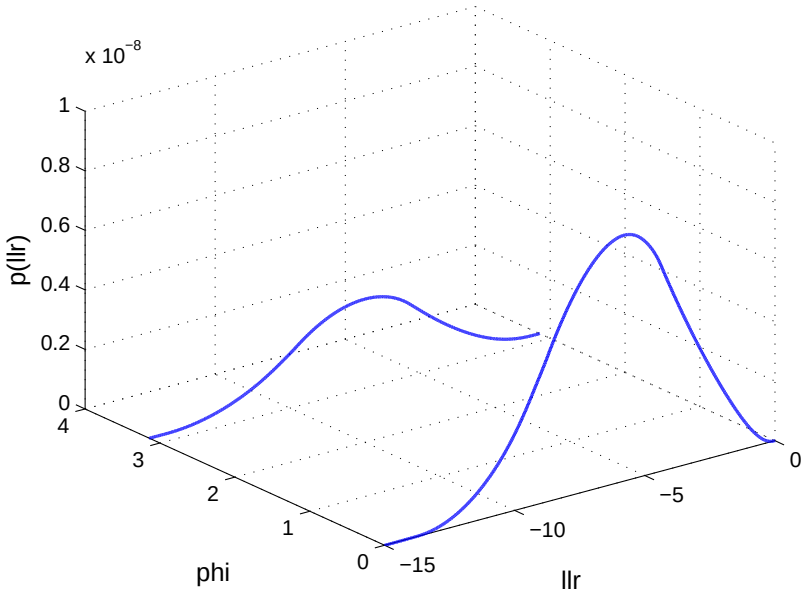

Figure 3.31: 2D convolution of one message node and one parity node

So far, the distributions tend to be consistent. According to the tanner graph, we still need convolute with one more distribution at message node:

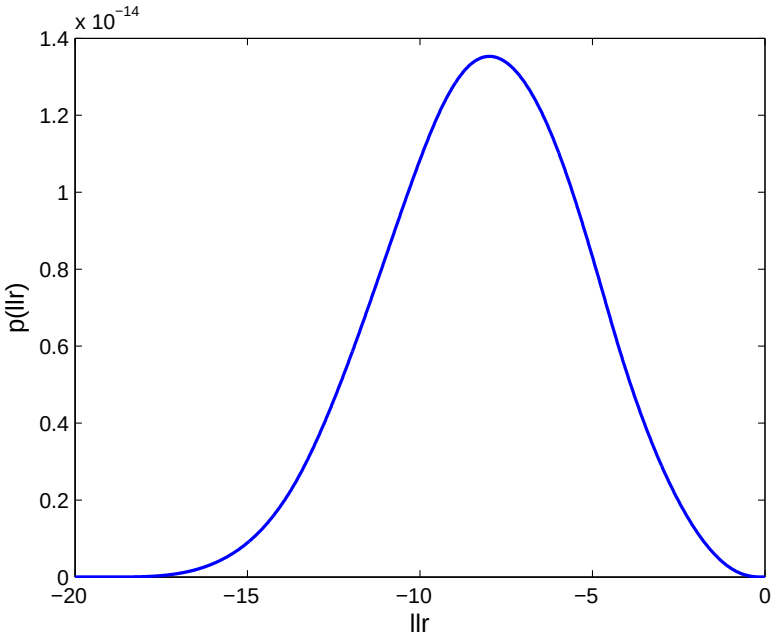

Figure 3.32: convolution when both distribution at $\phi = 0$

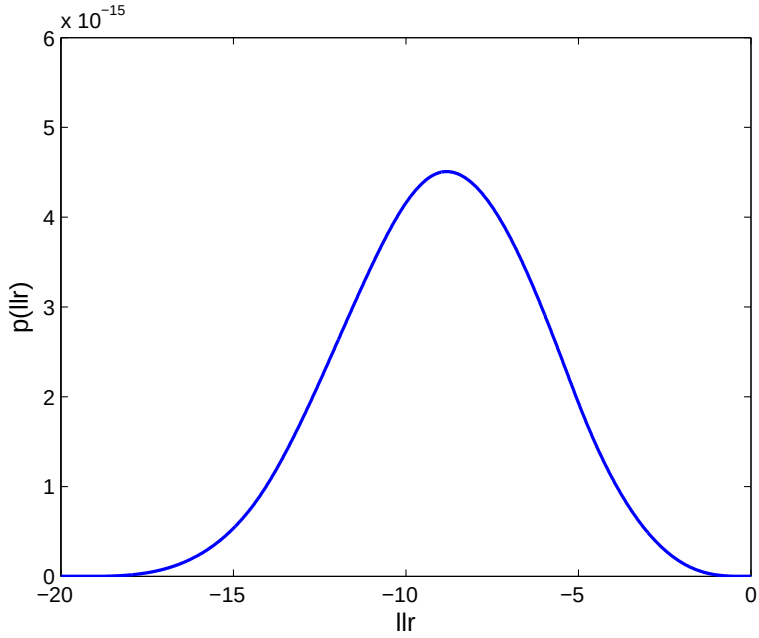

Figure 3.33: convolution when both distribution at $\phi = \pi$

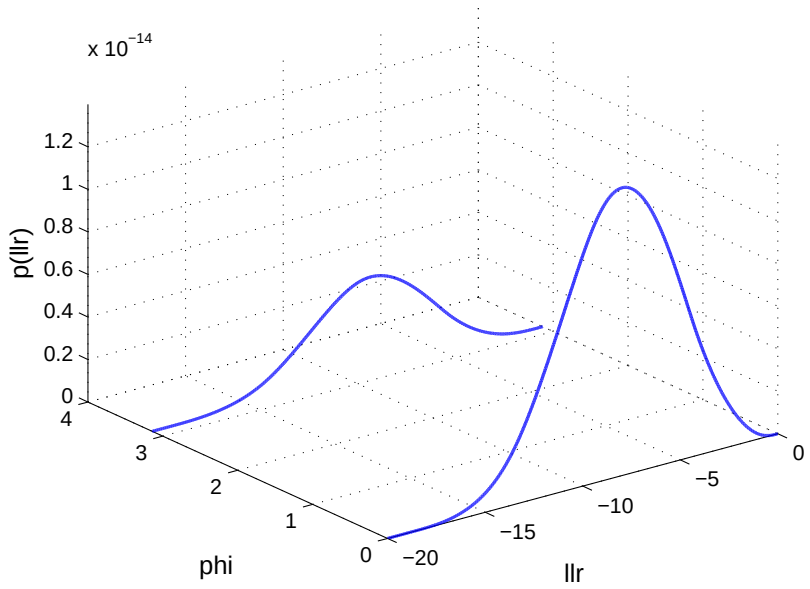

Figure 3.34: 2D convolution of one message node and one parity node

Up to the convolution of distributions at two parity nodes and two message nodes, the intermediate model is consistent and almost like the Gaussian distribution. The distribution converges very quickly.

3.3.4 1D calculation and simulation

Step 6: $x6 = \exp(x5)$
We apply the exponential operation on the convolution result.

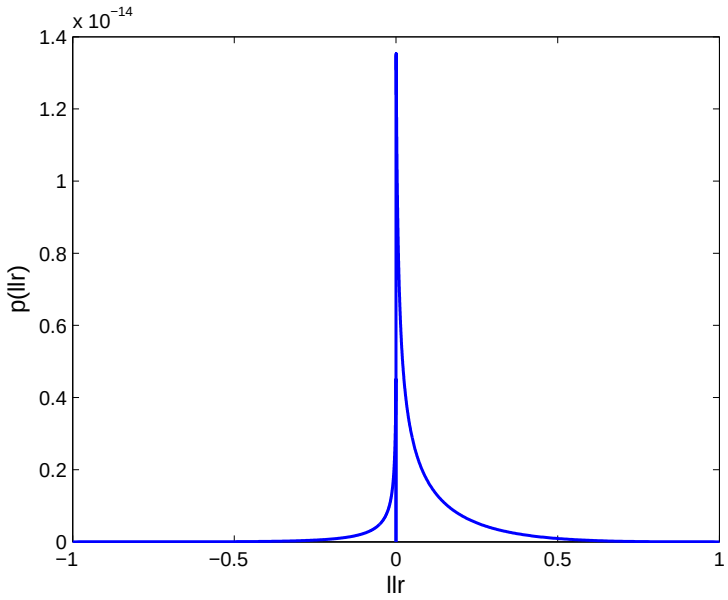

Figure 3.35: The Distribution after exponential

Though the intermediate model is consistent, however, after the exponential operation the distribution is non-consistent again. Therefore, we need more iteration to have further study. But the intermediate model could say that even though the initial distribution is non-Gaussian, it would turn to be consistent and Gaussian after several iteration.

Step 7: $y = \tanh^{-1}(x6)$

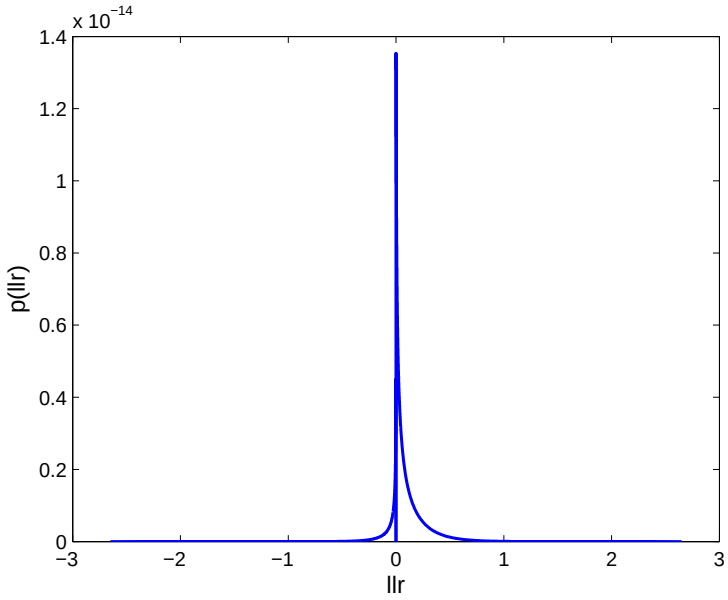

Figure 3.36: The Distribution after inverse hyperbolic tangent

The Fig. 3.36 is the final result and will be forwarded to the variable node side. Nevertheless, there are some errors, which may come from the rescale operation in Step 5. The discontinuity when LLR is equal to zero should be fixed in further studies.

Chapter 4

Conclusion

In this thesis, we focused on the density evolution procedure of the LDPC code. The outgoing messages are usually treated as Gaussians though this assumption cannot hold for the first several steps, since only the intrinsic information is forwarded at the beginning. We have dealt with distributions on variable node sides and check node sides, separately, while have paid more attention on the check node due to the complex nonlinear operation on this side. The thesis presents the simulation result of the distribution at check node side under the specific situation, only including five variable nodes and two check nodes. Nevertheless, the intermediate distribution of the LLRs will quickly turn to be consistent in the convolution, though the final result is still non-consistent.

Extensions of this work should pay attention to the discontinuity in the final information forwarded to variable node side. More iterations should be appied for obtaining the ideal consistent distribution. Meanwhile, we should deal with the abnormal distribution at $LLR = 0$, I guess it could come from the calculation error but need more proof. If it only comes from a calculation error, it could be easily corrected; otherwise, more research need to carry out to fix this problem.

Bibliography

[1] A. Shokrollahi, "LDPC codes: An introduction," Digital Fountain, Inc., Tech. Rep., 2003.

[2] X. Zhou, L. Song, and Y. Zhang, *Physical Layer Security in Wireless Communications*. Crc Press, 2013.

[3] A. Filip, "Physical-layer security," 2013.

[4] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *Information theory, IEEE Transactions on*, vol. 19, no. 4, pp. 471–480, 1973.

[5] J. Chen and P. Fossorier, "Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions," in *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, vol. 2. IEEE, 2002, pp. 1378–1382.

[6] W. Henkel, "Channel coding," Jacobs University Bremen, Tech. Rep., 2014.

[7] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 599–618, 2001.

[8] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 619–637, 2001.

[9] N. Islam, "Quantization aspects in LDPC decoding for key reconciliation methods for a reciprocal channel," 2014.