國　立　交　通　大　學

電信工程學系碩士班

碩士論文

MIMO-OFDM 系統之偵測與低密度同位檢查碼解碼

Detection and LDPC Codes Decoding for MIMO-OFDM Systems

研 究 生：黃冠榮

指導教授：吳文榕　博士

中　華　民　國　九十五 年　七　月

MIMO-OFDM 系統之偵測與低密度同位檢查碼解碼

Detection and LDPC Codes Decoding for MIMO-OFDM Systems

研 究 生：黃冠榮　　Student：Kuan-Jung Huang

指導教授：吳文榕 博士　　Advisor：Dr. Wen-Rong Wu

國 立 交 通 大 學

電信工程學系碩士班

碩 士 論 文

A Thesis

Submitted to Department of Communication Engineering

College of Electrical Engineering

National Chiao-Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

In

Communication Engineering

July 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年七月

MIMO-OFDM 系統之偵測與低密度同位檢查碼解碼

研究生：黃冠榮　　指導教授：吳文榕 博士

國立交通大學電信工程學系碩士班

中文摘要

本篇論文主要為研究多輸入多輸出（MIMO）正交分頻多工（OFDM）系統上之訊號偵測與低密度同位檢查碼之解碼。在低密度同位檢查碼的解碼上，我們使用了三種解碼演算法，包括 Normalized belief propagation（BP）based 演算法、Normalized a-posteriori probability（APP）based 演算法、和 Layer normalized BP based 演算法。這三種不同的解碼演算法，皆比低密度同位檢查碼標準的解碼演算法— Sum-product 演算法有更高的可實現性，同時在運算複雜度與效能的取捨上，都有不錯的表現。而在 MIMO 訊號的偵測上，我們使用了最小均方誤差（MMSE）與最大事後機率（MAP）偵測。其中，我們將 MMSE 偵測與軟性反對映結合以獲得軟性輸出（soft outputs），而 MAP 偵測則可直接由所接收到的訊號來求得軟性輸出。另外，為了降低 MAP 偵測本身龐大的計算量，我們在 MAP 偵測前預先使用了 list sphere decoding 演算法，以降低在 MIMO 訊號偵測上的運算複雜度。最後，我們在 IEEE 802.11n 與 IEEE 802.16e 系統上模擬，以驗證低密度同位檢查碼對系統效能上的改善，特別是使用 MAP 偵測 MIMO 訊號時，使用低密度同位檢查碼對系統效能會有相當大的改進。

Detection and LDPC Codes Decoding for MIMO-OFDM Systems

Student: Kuan-Jung Huang Advisor: Dr. Wen-Rong Wu

Department of Communication Engineering

National Chiao-Tung University

Abstract

In this thesis, we study signal detection and decoding of low-density parity-check (LDPC) codes for multi-input-multi-output (MIMO) orthogonal-frequency-division-multiplexing (OFDM) systems. Three types of LDPC codes decoders are investigated, the Normalized belief-propagation (BP) based algorithm, the Normalized a-posteriori probability (APP) based algorithm, and the Layered normalized BP based algorithms. These decoding algorithms are much simpler to implement than the standard LDPC codes decoding algorithm, namely the sum-product algorithm, and can achieve good tradeoff between decoding complexity and performance. For MIMO signal detection, we consider the minimum-mean-squared error (MMSE) and a maximum a-posteriori probability (MAP) detector. The MMSE detector is combined with a soft-bit demapper to obtain soft outputs, while the MAP detector is designed to have soft outputs directly. To redue the high computational inherent in the MAP detector, we apply an efficiency algorithm called the list sphere decoding. Simulations with IEEE 802.11n and IEEE 802.16e systems show that the LDPC codes decoder can effectively improve the system performance, particularly when it combined with the MAP detector.

誌謝

本篇論文得以完成，首先要感謝的就是我的指導老師　吳文榕教授，我至今仍記得：兩年前，老師在學生名額已滿的情況下，仍然不吝增加自己的負擔，加收了我這樣的一個學生，讓我有幸在老師的實驗室內完成我兩年的學業與論文研究。在我研究所求學期間，老師亦給我論文研究詳盡的指導，使我論文研究之研究方向不會偏離正軌，而老師細心、嚴謹的求學態度更使我受益匪淺。這裡，真的是非常的感謝老師。

其次，我要感謝李俊芳學長、楊華龍學長、李彥文學長、許兆元學長和李峰宇學長，在我的研究及課業學習上所提供的的指導與協助，且同時感謝寬頻通訊與訊號處理實驗室所有同學與學弟妹們的幫忙。其中，特別感謝士琦、星宇、宛儀和坤浩，以及我大學的摯友明哲，本篇論文可以說是由大家所完成的，在我的論文中：11n 的系統是士琦所提供的，16e 的介面則是由星宇整合的，而 11n 和 16e 模擬所需的通道模型則是宛儀所建立的，甚至我在 LDPC Codes 上的疑惑，都是明哲不厭其煩的解釋，我才得以明白，如果沒有大家的協助，我是不可能完成本篇論文的。最後致上我最深的感謝給我的父母和姊姊，他們給予我精神上的鼓勵和經濟上的支持，使我無後顧之憂順利完成研究所的碩士學位，謝謝你們。

民國九十五年七月

研究生黃冠榮謹識於交通大學

內容目錄

表目錄

圖目錄

第1章 簡介

1948 年Shannon發表了現代通訊理論最重要的論文“A Mathematical Theory of Communication [1]”，精準地預測所有通訊系統的基本極限，並開創了一個全新的數學領域：消息理論（Information Theory）。他提出對於任何傳輸速率小於或等於通道容量（capacity）的情況下，必定存在一種編碼方式，可以達成任意小的錯誤率。雖然他並未具體指出如何設計此種錯誤更正碼（error correction code），但卻提供了相當寶貴的依循法則，也為通訊工程指引了一個明確的目標。目前較為熟知的錯誤更正碼包括線性區段碼（Linear Block Codes）[2]、漢明碼（Hamming Codes）[3]、摺積碼（Convolutional Codes）[4]，以及里德所羅門碼（Reed-Solomon Codes）[5]。近 10 年來，則以渦輪碼（Turbo Codes）[6]和低密度同位檢查碼（LDPC Codes）[7]最受重視。LDPC Codes在 1962 年由MIT的Robert Gallager發明，當時卻因被認為複雜度過高而被遺忘將近 30 年，直至近年來被重新研究後才發現其優越的性能。

LDPC Codes是一種特殊的線性區段碼，藉著定義一個同位檢查矩陣（parity check matrix），我們得以有系統地產生碼字（codeword），並規範訊息位元（message bits）之間的關係。在LDPC Codes中的同位檢查矩陣為一稀疏矩陣（sparse matrix），因此稱之為低密度（Low Density）。每一個LDPC Codes的同位檢查矩陣皆可以Tanner Graph[8]表示。而LDPC Codes的解碼方式則是在Tanner Graph上利用Sum-product演算法 [9]進行check node和bit node的訊息可靠度交換。

LDPC Codes 運作原理相似於 Turbo Codes，藉由重複遞迴（iterative）的處理方式來逼近沈農極限（Shannon limit）。然而 Turbo Codes 的碼尺寸（code size）過大，因此在解碼過程經歷數次遞迴後會造成延遲時間過長，因此無法適用於WLAN、即時語音通訊或需要即時數據處理等應用。相較之下，由於 LDPC Codes的行為容易分析、error floor 較低、硬體實現度高以及較高的解碼速度（在 LDPC Codes 中主要是記憶體處理，而 Turbo Codes 則完全由運算處理實現）。此外，

LDPC Codes 專利已經過期，所以很多公司都可以使用而不必付費。

目前 LDPC Codes 在 IEEE 802.11n、IEEE 802.15.3a 以及 IEEE 802.16e 的技術提案中皆被熱烈討論，而下一代衛星數位視訊廣播標準（DVB-S2）也決議以 LDPC Codes 取代 Turbo Codes。在未來，以 LDPC Codes 取代 Turbo Codes 的後勢相當看好。

多輸入多輸出(Multiple-input Multiple-output：MIMO)系統，該技術最早是由 Marconi 於 1908 年提出的，它利用多天線來抑制通道衰落。根據收發兩端天線數量，相對於普通的 SISO（Single-Input Single-Output）系統，MIMO 還可以包括 SIMO（Single-Input Multiple-Output）系統和 MISO（Multiple-Input Single-Output）系統。原則上，通道容量隨著天線數量的增大而線性增大。也就是說可以利用 MIMO 系統倍數地提高無線通道容量，在不增加頻寬和天線發送功率的情況下，頻譜利用率可以倍數地提高。

MIMO的核心概念為利用多根發射天線與多根接收天線所提供之空間自由度提升傳輸速率與改善通訊品質；它主要有兩種功能形式：一為空間多工（spatial multiplex），另一為空間分集（spatial diversity）。前者是在發射端利用多根天線傳送不同資料序列，並在接收端利用多根天線的空間自由度將該組資料序列分別解出。經由此一程序，在發射端與接收端之間彷彿形成一組虛擬的平行空間通道，可在同一時間、同一頻段，以同一功率傳送多個資料序列。如此一來，整體系統的有效資料傳輸率便可以在不增加任何通訊資源的前題下提升數倍。而後者是利用發射或接收端的多根天線所提供的多重傳輸途徑來對抗通道衰落（fading）的影響；所謂分集意即多重選擇性，它可由多個獨立的傳輸途徑中選擇或組合出衰落現象較輕微的接收訊號，以維持穩定的鏈路品質。空間多工接收機的演算法主要有貝爾實驗室的BLAST演算法、ZF演算法、MMSE演算法 [13]、ML演算法等 [20]。而空間分集主要代表便是空時區塊編碼(Space-Time Block Coding, STBC)，它於發射端將待傳送之資料符元（data symbol）在空間與時間上作預前編碼，產生適當的冗餘（redundancy），並在接收端經由簡易的處理將此冗餘轉

化為「分集增益」(diversity gain)。

本篇論文主要是在討論LDPC Codes的解碼器設計，與MIMO系統的偵測（detection)。在LDPC Codes的解碼器設計上，由於Sum-product演算法的複雜性過高，不利於硬體的設計與實現，在此我們使用了三種以Sum-product演算法為基礎而簡化的LDPC Codes解碼演算法，並在 802.11n與 802.16e的系統上，模擬比較此三種LDPC Codes解碼演算法的效能。而在MIMO系統的偵測上，則討論現存MIMO偵測上較常見的MMSE演算法，與以ML演算法為基礎的MAP偵測[20]，再結合上述三種以Sum-product演算法為基礎而簡化的LDPC Codes解碼演算法，並在 802.11n的系統上模擬比較此兩種MIMO偵測演算法的效能。

本篇論文的結構如下: 第二章將介紹LDPC Codes，包括編碼的原理，與解碼演算法，其中包含Sum-product演算法，與以Sum-product演算法為基礎而簡化的Normalized belief propagation based（Normalized BP-based）演算法 [10]、Normalized a-posteriori probability based（Normalized APP-based）演算法 [11]、Layered belief propagation（Layered BP）演算法 [12]。第三章則簡介MIMO-OFDM系統，包括此系統的數學模型建立、MIMO MMSE偵測演算法 [13]，軟式反對映（soft demapping）[14]，和LDPC Codes在 802.11n[15][16]與 802.16e[18][19]的系統上的應用。第四章則討論以MAP演算法為基礎的偵測，並將之與List Sphere decoding演算法 [22]結合，以降低MAP偵測的運算複雜度。最後總結在第五章中。

第2章 低密度同位檢查碼（LDPC Codes）

低密度同位檢查碼LDPC Codes（Low-Density Parity-Check Codes）[7]原本是由Robert Gallager在 1962 年所發明的，這是一種使用長段的線性區段碼（Linear Block Codes）。由於當時電腦能力不足以處理複雜性運算，因此使人們淡忘許久，直到 1995 年由Mackey與Neal重新發展出Tanner Graph[8]，在解碼時使用兩個狀態的同位檢查格（parity check trellis），使得解碼器容易實現，並使其效能接近沈農極限（Shannon limit），加上近幾年來VLSI技術的快速發展，亦使得LDPC Codes又逐漸被人們所廣為討論。

2.1 編碼器（Encoder）

LDPC Codes 的編碼方式為由產生矩陣（generator matrix）G 乘上訊息向量（information vector）$\vec{U}$ 而產生碼字向量（codeword vector）$\vec{V}$ ，即$\vec{U}G=\vec{V}$。由於 LDPC Codes 亦為線性區段碼（Linear Block Codes），故須符合碼字向量$\vec{V}$ 乘上同位檢查矩陣（parity check matrix）H_p 後等於零的規則，亦即 $H_p\vec{V}^T=0$ ，所以可由下列方法得到產生矩陣G和碼字向量$\vec{V}$。假設一 LDPC Codes 以(N,K)的線性區段碼表示，則其中有 $M=N-K$ 個檢查位元（parity check bits），K 個訊息位元（message bits）與訊息向量$\vec{U}$ 相同。則：

$$H_p\vec{V}^T=H_p\left(\vec{U}G\right)^T=0 \tag{2.1}$$

$$\Rightarrow H_pG^T=0 \tag{2.2}$$

。由於將同位檢查矩陣H_p中的任意兩列交換或進行 MOD2 運算，所得之新同位檢查矩陣仍可滿足(2.2)式，故可將原來的同位檢查矩陣H_p使用高斯消去法

（Gaussian elimination）推導至(2.3)式：

$$H_p' = \left[P_{M\times K} \vdots I_{M\times M} \right] \tag{2.3}$$

其中$P_{M\times K}$為同位檢查部（parity check part），$I_{M\times M}$為$M\times M$的單位矩陣（identity matrix）。而其所對應的產生矩陣G為：

$$G = \left[I_{K\times K} \vdots P_{K\times M}^T \right] \tag{2.4}$$

，將(2.3)式與(2.4)式代入(2.2)式中驗證：

$$H_p' G^T = \left[I_{M\times M} \vdots P_{M\times K} \right] \begin{bmatrix} P_{M\times K} \\ \cdots \\ I_{K\times K} \end{bmatrix} = P_{M\times K} \oplus P_{M\times K} = 0 \tag{2.5}$$

。一般說來，產生矩陣G通常不為稀疏矩陣（sparse matrix），故編碼時的運算複雜度，會隨碼字長度（codeword length）N呈N^2級數增加。

2.2 圖形解碼

目前LDPC Codes有多種解碼的演算法，其中包括：Majority-logic（MLG）解碼、Bit-flipping（BF）解碼、Sum-product演算法（SPA）[9]，而Sum-product演算法又有其它別名：Belief propagation演算法（BPA）、Message Passing algorithm（MPA）。在上述LDPC Codes的解碼演算法中，以Sum-product演算法的效能最好，有最低的位元錯誤率（bit error rate），然而其運算複雜度亦最高，硬體實現上也最困難。因此，吾人使用了三種以Sum-product演算法為基礎而修正的演算法：Normalized belief propagation based（Normalized BP-based）演算法 [10]、Normalized a-posteriori probability based（Normalized APP-based）演算法 [11]、Layered belief propagation（Layered BP）演算法 [12]，以降低運算複雜度，增加硬體的可實現性。

2.2.1 Tanner Graph

在 LDPC Codes 中，可將一 $M \times N$ 的同位檢查矩陣 H_p 分成兩個部份，這兩個部份分別包含 check node 和 bit node。第一個部份是把同位檢查矩陣 H_p 的每一列（row）看成一個 check node，故總共有 M 個 check node。第二個部份是把每一行（column）看成一個 bit node，也就是碼字的長度，相當於有 N 個 bit node。由於 LDPC Codes 解碼的過程要符合 $H_p\vec{V}^T = 0$，所以連結到同一個 check node 的所有 bit node 均要滿足 $H_p\vec{V}^T = 0$ 的關係式，也就是說只有在該列上元素（element）為“1”的 bit，其 bit node 才有連結到該列所代表的 check node。舉例來說，若 $H_p = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$，則 Tanner Graph 可表示為下圖：

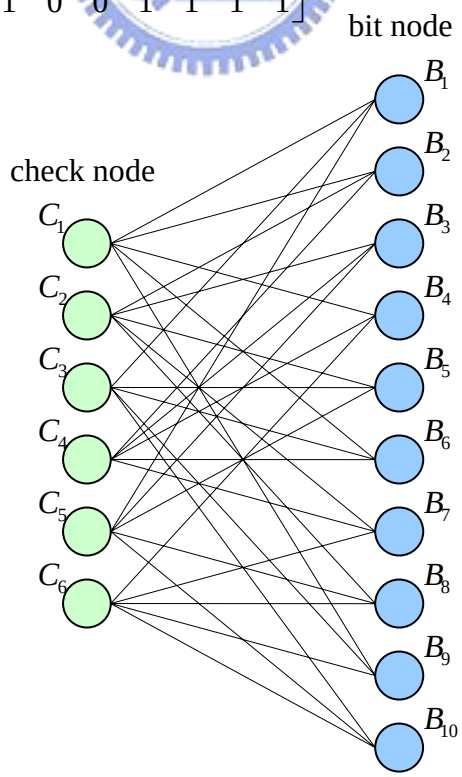


圖 2-1 Tanner Graph

2.2.2 圖形解碼概念

由圖 2-2 的解碼器架構圖可知 LDPC Codes 的解碼過程是應用到 Message Passing 的概念，由 bit node 及 check node 這兩端互相算出機率再傳送給另一端。在計算由一特定 bit node 到某一 check node 的機率，是由其它連結到該 bit node 所有 check node，及連結到此 bit node 的 another node 所決定的。同樣的，要計算由一特定的 check node 到某一 bit node 的機率時，亦是由其它連結到該 check node 所有 bit node 來求出。而在最後要求某一 bit 的機率，則是由所有連結到該 bit node 的 check node 及 another node 決定的。下節將討論的其解碼演算過程。

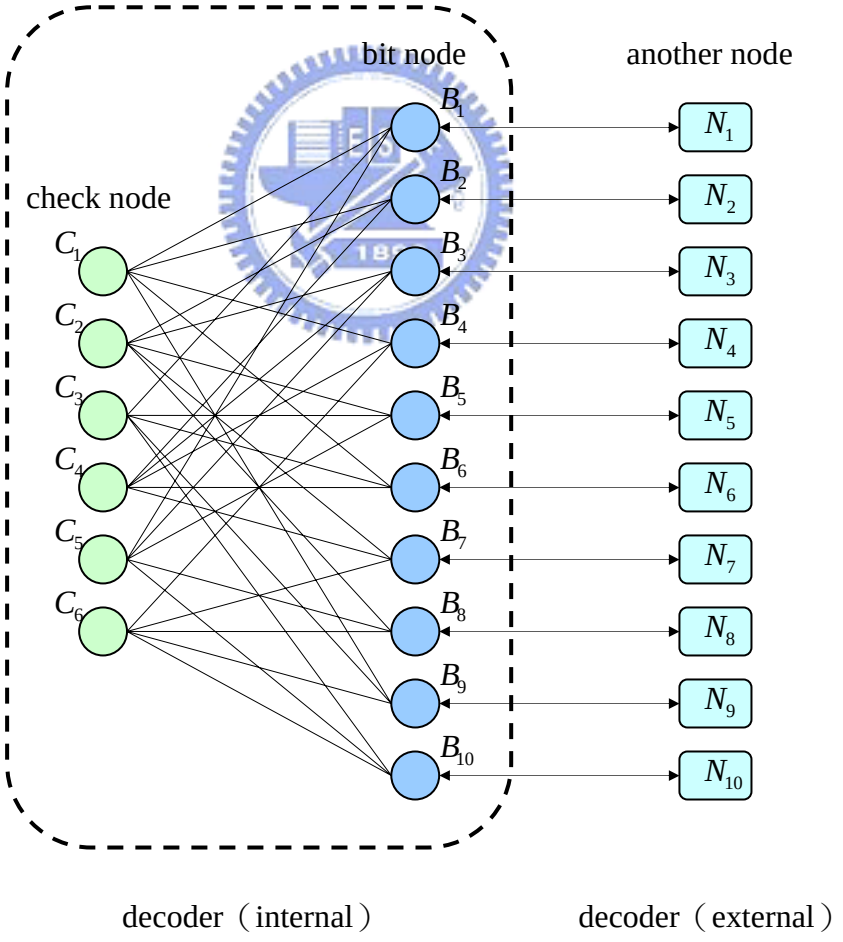


圖 2-2 解碼器架構圖

2.3 Sum-Product 演算法

2.3.1 Bit node 至 Check node 機率資訊

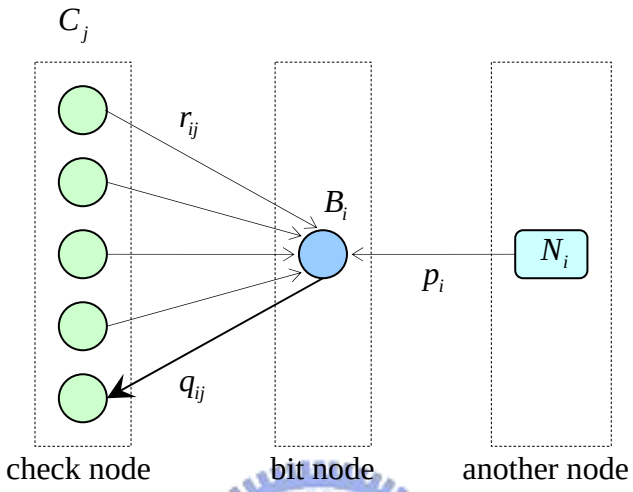


圖 2-3 bit node 至 check node 機率資訊

如圖 2-3 所示，其中 p_i 表示由 another node N_i 傳送至 bit node B_i 的機率資訊，r_{ij} 表示由 check note C_j 傳送至 bit node B_i 的機率資訊，q_{ij} 表示由 bit node B_i 傳送至 check note C_j 的機率資訊。若 bit node B_i 與 K 個 check node 及 another node N_i 相連結，且彼此獨立（independent），則：

$$P\left(q_{ij}=\xi\right)=P\left(p_i=\xi\right)\prod_{j'\in M(i)\setminus\{j\}}P\left(r_{ij'}=\xi\right) \tag{2.6}$$

，其中 ξ 等於1或0，$M\left(i\right)$ 表示同位檢查矩陣 H_p 之第 i 行為“1”之列的集合，而 $M\left(i\right)\setminus\left\{j\right\}$ 則表示在集合 $M\left(i\right)$ 中，不包含第 j 個元素之子集合。

為了簡化運算，我們使用Log-Likelihood ratio（LLR），則依據LLR之定義：

$$LLR\left(a\right)=\log\frac{P\left(a=1\right)}{P\left(a=0\right)} \tag{2.7}$$

，可求得 $LLR(r_{ij})$、$LLR(p_i)$ 與 $LLR(q_{ij})$，並將之代入 (2.6) 式可得：

$$LLR(q_{ij}) = LLR(p_i) + \sum_{j' \in M(i) \setminus \{j\}} LLR(r_{ij'}) \tag{2.8}$$

，(2.8) 式即為以 LLR 形式所表示之 bit node 至 check node 機率資訊之計算式。

2.3.2 Check node 至 Bit node 機率資訊

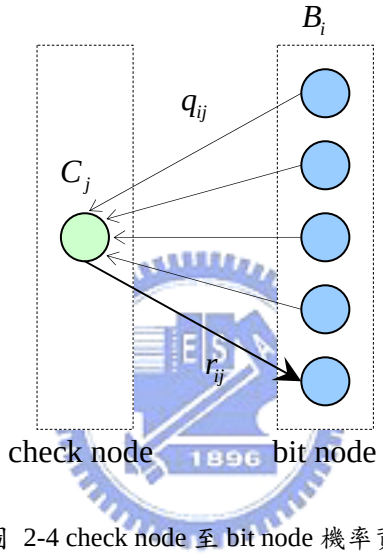


圖 2-4 check node 至 bit node 機率資訊

如圖 2-4 所示，check node C_j 與 K 個 bit node 相連結且彼此獨立。由於解碼的過程需符合 $H_p\vec{V}^T = 0$ 的關係式，所以連結到同一個 check node 的所有 bit node 要滿足：

$$B_1 \oplus B_2 \oplus B_3 \cdots \oplus B_{i-1} \oplus B_i \oplus B_{i+1} \cdots \oplus B_K = 0 \tag{2.9}$$

，則由 (2.9) 式，可推得：

$$\begin{aligned} P(r_{ij} = 1) &= P(B_1 \oplus B_2 \oplus B_3 \cdots \oplus B_{i-1} \oplus B_{i+1} \cdots \oplus B_K = 1) \\ P(r_{ij} = 0) &= P(B_1 \oplus B_2 \oplus B_3 \cdots \oplus B_{i-1} \oplus B_{i+1} \cdots \oplus B_K = 0) \end{aligned} \tag{2.10}$$

。為了求出 (2.10) 式的通式，將 $K = 2$ 代入，並假設 $P(B_1 = 1) = a_1$、$P(B_2 = 1) = a_2$

可求得下式：

$$\begin{aligned} P(B_1 \oplus B_2 = 1) &= a_1(1-a_2) + a_2(1-a_1) \\ P(B_1 \oplus B_2 = 0) &= a_1 a_2 + (1-a_1)(1-a_2) \end{aligned} \tag{2.11}$$

，我們將(2.11)式進一步化簡成下式：

$$\begin{aligned} P(B_1 \oplus B_2 = 1) &= \frac{1-(1-2a_2)(1-2a_1)}{2} = \frac{1-\prod_{i=1}^{2}(1-2a_i)}{2} \\ P(B_1 \oplus B_2 = 0) &= \frac{1+(1-2a_2)(1-2a_1)}{2} = \frac{1+\prod_{i=1}^{2}(1-2a_i)}{2} \end{aligned} \tag{2.12}$$

，若$K = n$時亦成立，可推得：

$$\begin{aligned} P(B_1 \oplus B_2 \oplus B_3 \cdots \oplus B_{n-1} = 1) &= \frac{1-\prod_{i=1}^{n-1}(1-2a_i)}{2} = M_{n-1} \\ P(B_1 \oplus B_2 \oplus B_3 \cdots \oplus B_{n-1} = 0) &= \frac{1+\prod_{i=1}^{n-1}(1-2a_i)}{2} = 1-M_{n-1} \end{aligned} \tag{2.13}$$

，則：

$$2M_{n-1} + 1 = \prod_{i=1}^{n-1}(1-2a_i) \tag{2.14}$$

，假設在$K = n+1$時，$P(B_n = 1) = a_n$：

$$\begin{aligned} P(B_1 \oplus B_2 \oplus B_3 \cdots \oplus B_{n-1} \oplus B_n = 1) &= a_n(1-M_{n-1}) + M_{n-1}(1-a_n) \\ P(B_1 \oplus B_2 \oplus B_3 \cdots \oplus B_{n-1} \oplus B_n = 0) &= a_n M_{n-1} + (1-a_n)(1-M_{n-1}) \end{aligned} \tag{2.15}$$

，由(2.12)式及(2.14)式，可將(2.15)式推導至下式：

$$\begin{aligned} P(B_1 \oplus B_2 \cdots \oplus B_n = 1) &= \frac{1-(1-2a_n)(1-2M_{n-1})}{2} = \frac{1-\prod_{i=1}^{n}(1-2a_i)}{2} \\ P(B_1 \oplus B_2 \cdots \oplus B_n = 0) &= \frac{1+(1-2a_n)(1-2M_{n-1})}{2} = \frac{1+\prod_{i=1}^{n}(1-2a_i)}{2} \end{aligned} \tag{2.16}$$

，經由歸納法證明，我們由(2.16)式之結果，來推得(2.10)式之通式：

$$P\left(r_{ij}=1\right)=P\left(B_1\oplus B_2\cdots\oplus B_{i-1}\oplus B_{i+1}\cdots\oplus B_K=1\right)=\frac{1-\prod_{i'\in L(j)\backslash\{i\}}\left(1-2Q_{i'j}\right)}{2}$$
$$P\left(r_{ij}=0\right)=P\left(B_1\oplus B_2\cdots\oplus B_{i-1}\oplus B_{i+1}\cdots\oplus B_K=0\right)=\frac{1+\prod_{i'\in L(j)\backslash\{i\}}\left(1-2Q_{i'j}\right)}{2} \quad (2.17)$$

，其中 $Q_{i'j}=P\left(q_{i'j}=1\right)$，$L(j)$ 表示同位檢查矩陣 H_p 之第 j 列為“1”之行的集合，而 $L(j)\backslash\{i\}$ 則表示在集合 $L(j)$ 中，不包含第 i 個元素之子集合。

依據 (2.7) 式 LLR 之定義，(2.17) 式可進一步簡化為：

$$LLR\left(r_{ij}\right)=\log\frac{P\left(r_{ij}=1\right)}{P\left(r_{ij}=0\right)}=\log\frac{1-\prod_{i'\in L(j)\backslash\{i\}}\left(1-2Q_{i'j}\right)}{1+\prod_{i'\in L(j)\backslash\{i\}}\left(1-2Q_{i'j}\right)} \quad (2.18)$$

，接著，我們求出下兩式：

$$LLR\left(q_{i'j}\right)=\log\frac{Q_{i'j}}{1-Q_{i'j}}\Rightarrow e^{LLR\left(q_{i'j}\right)}=\frac{Q_{i'j}}{1-Q_{i'j}} \quad (2.19)$$

$$\tanh\left(x\right)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}\Rightarrow\tanh\left(\frac{x}{2}\right)=\frac{e^{x}-1}{e^{x}+1} \quad (2.20)$$

，並將 $x=LLR\left(q_{i'j}\right)$ 代入 (2.20) 式中：

$$\frac{\frac{Q_{i'j}}{1-Q_{i'j}}-1}{\frac{Q_{i'j}}{1-Q_{i'j}}+1}=2Q_{i'j}-1=\tanh\left(\frac{1}{2}LLR\left(q_{i'j}\right)\right) \quad (2.21)$$

，再把 (2.21) 式之結果代入 (2.18) 式內：

$$LLR\left(r_{ij}\right)=\log\frac{1-\left(-1\right)^{\left|L(j)\right|-1}\prod_{i'\in L(j)\backslash\{i\}}\tanh\left(\frac{1}{2}LLR\left(q_{i'j}\right)\right)}{1+\left(-1\right)^{\left|L(j)\right|-1}\prod_{i'\in L(j)\backslash\{i\}}\tanh\left(\frac{1}{2}LLR\left(q_{i'j}\right)\right)} \quad (2.22)$$

，其中 $\left|L(j)\right|$ 表示同位檢查矩陣 H 之第 j 列為“1”之行的個數。

$$\tanh^{-1}\left(y\right)=\frac{1}{2}\log\frac{1+y}{1-y}\ \Rightarrow\ -2\tanh^{-1}\left(y\right)=\log\frac{1-y}{1+y} \quad (2.23)$$

最後，我們將 $y=(-1)^{|L(j)|-1}\prod_{i'\in L(j)\setminus\{i\}}\tanh\left(\frac{1}{2}LLR\left(q_{i'j}\right)\right)$ 代入 (2.23) 式中，可求得：

$$LLR\left(r_{ij}\right)=2\times(-1)^{|L(j)|}\times\tanh^{-1}\left(\prod_{i'\in L(j)\setminus\{i\}}\tanh\left(\frac{1}{2}LLR\left(q_{i'j}\right)\right)\right) \tag{2.24}$$

，(2.24) 式即為以 LLR 形式所表示之 check node 至 bit node 機率資訊之計算式。

2.3.3 Bit node 事後（a-posteriori）機率資訊

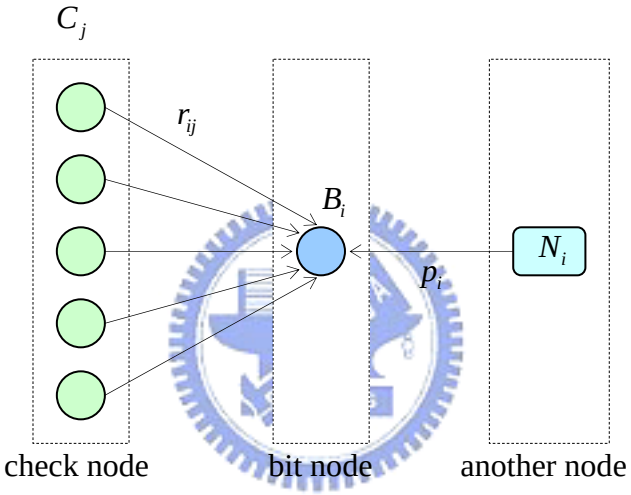


圖 2-5 事後機率資訊

如圖 2-5 所示，由於碼字中的位元 B_i 其“0”與“1”的機率，是由所有與 bit node B_i 連結的 check node 及 another node N_i 所決定，所以我們可推得：

$$P\left(B_i=\xi\right)=P\left(p_i=\xi\right)\prod_{j\in M(i)}P\left(r_{ij}=\xi\right) \tag{2.25}$$

，再依照 (2.7) 式，將 (2.25) 式以 LLR 的形式表示：

$$LLR\left(B_i\right)=LLR\left(p_i\right)+\sum_{j\in M(i)}LLR\left(r_{ij}\right) \tag{2.26}$$

，(2.26) 式即為以 LLR 形式所表示之 bit node B_i 事後機率資訊之計算式。

最後再由(2.26)式所得之結果進行判決（decision），若$LLR(B_i)\geq 0$，則表示經過解碼後位元B_i判決為“1”，反之若$LLR(B_i)<0$，則表示經過解碼後位元B_i判決為“0”，我們可用下式表示：

$$B_i=\begin{cases}1 & LLR(B_i)\geq 0\\ 0 & LLR(B_i)<0\end{cases} \tag{2.27}$$

2.3.4 Sum-Product 演算法解碼流程

第一步：初始化（initialize），設定最大遞迴次數“k_{Max}”，並令一開始的 check node 至 bit node 之“1”與“0”的機率資訊是相等的：

$$LLR^{(0)}(r_{ij})=\log\frac{0.5}{0.5}=0 \tag{2.28}$$

，其中，上標“(k)”內的數字“k”，表示為第k次遞迴所得之結果。

第二步：計算 bit node 至 check node 之機率資訊：

$$LLR^{(k)}(q_{ij})=LLR(p_i)+\sum_{j'\in M(i)\backslash\{j\}}LLR^{(k-1)}(r_{ij'}) \tag{2.29}$$

第三步：計算 check node 至 bit node 之機率資訊：

$$LLR^{(k)}(r_{ij})=2\times(-1)^{|L(j)|}\times\tanh^{-1}\left(\prod_{i'\in L(j)\backslash\{i\}}\tanh\left(\frac{1}{2}LLR^{(k)}(q_{i'j})\right)\right) \tag{2.30}$$

第四步：計算 bit node B_i 事後機率資訊，並進行判決：

$$LLR^{(k)}(B_i)=LLR(p_i)+\sum_{j\in M(i)}LLR^{(k)}(r_{ij}) \tag{2.31}$$

$$B_i^{(k)}=\begin{cases}1 & LLR^{(k)}(B_i)\geq 0\\ 0 & LLR^{(k)}(B_i)<0\end{cases} \tag{2.32}$$

第五步：重複遞迴，直到解出的碼字符合$H_p\vec{V}^T=0$，或$k=k_{Max}$。

2.4 Normalized BP-based 演算法

2.4.1 UMP（Uniformly Most Powerful）BP-based 演算法

由於(2.24)式，check node 至 bit node 之機率資訊的計算式中，內含 tanh 及 $\tanh^{-1}$ 的函式，故在硬體的設計上極為困難，為了降低硬體的複雜度，我們使用了其它的函式來逼近(2.24)式，以方便電路的實現。

首先我們引進兩個式子：

$$\prod_i a_i = \left(\prod_i \operatorname{sgn}(a_i)\right)\exp\left(\sum_i \log\left(|a_i|\right)\right) \tag{2.33}$$

$$\Psi(x) = -\log\left(\tanh\left(\frac{x}{2}\right)\right), \quad x>0 \tag{2.34}$$

。將(2.33)式及(2.34)式代入(2.24)式中，可推導出：

$$\begin{aligned} LLR(r_{ij}) &= 2(-1)^{|L(j)|}\tanh^{-1}\left(\prod_{i'\in L(j)\setminus\{i\}}\tanh\left(\frac{LLR(q_{i'j})}{2}\right)\right) \\ &= 2(-1)^{|L(j)|}\tanh^{-1}\left(\left(\prod_{i'\in L(j)\setminus\{i\}}\operatorname{sgn}\left(\tanh\left(\frac{LLR(q_{i'j})}{2}\right)\right)\right)\right. \\ &\qquad \left.\times\exp\left(\sum_{i'\in L(j)\setminus\{i\}}\log\left(\left|\tanh\left(\frac{LLR(q_{i'j})}{2}\right)\right|\right)\right)\right) \\ &= (-1)^{|L(j)|}\times\prod_{i'\in L(j)\setminus\{i\}}\operatorname{sgn}\left(LLR(q_{i'j})\right)\times\Psi\left(\sum_{i'\in L(j)\setminus\{i\}}\Psi\left(\left|LLR(q_{i'j})\right|\right)\right) \end{aligned} \tag{2.35}$$

由於函數 $y=\operatorname{sgn}|x|\times\Psi(|x|)=-\operatorname{sgn}|x|\times\log\left(\tanh\left(\frac{|x|}{2}\right)\right)$ 有兩個特性：

第一個特性為當 $|x|$ 值越小，對應的 $|y|$ 值會越大，反之亦然，如圖 2-6 之曲線圖。

第二個特性為 $\Psi(x)$ 即為自己本身的反函數，如下式所示：

$$\Psi(\Psi(x))=x,\ \ x>0 \ \Rightarrow \operatorname{sgn}(x)\times\Psi\left(\operatorname{sgn}(x)\times\Psi(|x|)\right)=x \tag{2.36}$$

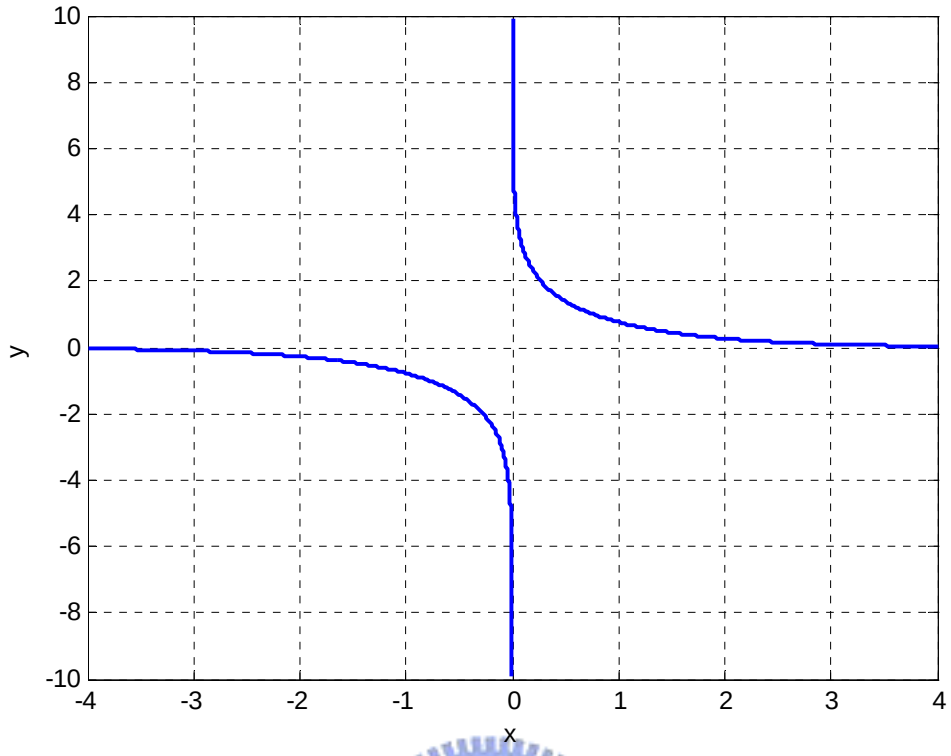


圖 2-6 $y = \mathrm{sgn}|x| \times \Psi(|x|)$ 函數圖

由函數 $\Psi(x)$ 的第一個特性，可知在 (2.35) 式中的 $\sum_{i' \in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)$ 之值會與最大的 $\Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)$ 之值近似，而最大的 $\Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)$ 又會被最小的 $\left|LLR\left(q_{i'j}\right)\right|$ 所決定，故我們可推得下式：

$$\sum_{i' \in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right) \approx \max_{i' \in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right) = \Psi\left(\min_{i' \in L(j)\setminus\{i\}} \left|LLR\left(q_{i'j}\right)\right|\right) \quad (2.37)$$

，又因為函數 $\Psi(x)$ 的第二個特性，我們可將 (2.37) 式之結果代入 $\Psi\left(\sum_{i' \in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)\right)$ 中，並近似為 (2.38) 式：

$$\begin{aligned} &\Psi\left(\sum_{i' \in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)\right) \\ &\approx \Psi\left(\Psi\left(\min_{i' \in L(j)\setminus\{i\}} \left|LLR\left(q_{i'j}\right)\right|\right)\right) = \min_{i' \in L(j)\setminus\{i\}} \left|LLR\left(q_{i'j}\right)\right| \end{aligned} \quad (2.38)$$

因此(2.24)式，check node 至 bit node 之機率資訊的計算式可再做進一步的化簡，並以(2.39)式近似之：

$$LLR\left(r_{ij}\right) \approx$$
$$LLR'\left(r_{ij}\right) = \left(-1\right)^{\left|L(j)\right|} \times \prod_{i' \in L(j)\backslash\{i\}} \operatorname{sgn}\left(LLR\left(q_{i'j}\right)\right) \times \min_{i' \in L(j)\backslash\{i\}} \left|LLR\left(q_{i'j}\right)\right| \tag{2.39}$$

，藉由(2.39)式的結果，我們可把原本(2.24)式複雜的 tanh 及 $\tanh^{-1}$ 函數簡化為 min 函數，以利硬體電路的實現。

2.4.2 正規化參數（Normalization factor）

若列權重（row weight（列之元素為"1"的個數））等於 ρ，在比較(2.24)式與(2.39)式後，我們可發現，在(2.24)式的 $\sum_{i' \in L(j)\backslash\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)$ 的項次為1，而(2.39)式的 $\max_{i' \in L(j)\backslash\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)$ 的項次為 $\rho - 1$，兩式之間相差 $\rho - 2$ 個項。故當 LDPC Codes 同位檢查矩陣 H_p 其碼字長度短，列權重小時，UMP BP-based 演算法之效能與 Sum-product 演算法並不會有太大的間距（gap），但隨著碼字長度變長，列權重亦增加時，兩者之間將會有 1dB 以上的間距。

藉由(2.35)式與(2.39)式，可觀察到 $LLR\left(r_{ij}\right)$ 與 $LLR'\left(r_{ij}\right)$ 之間有兩樣特性：

特性一、$LLR\left(r_{ij}\right)$ 與 $LLR'\left(r_{ij}\right)$ 恆同時為正或同時為負：

$$\operatorname{sgn}\left(LLR\left(r_{ij}\right)\right) = \operatorname{sgn}\left(LLR'\left(r_{ij}\right)\right) \tag{2.40}$$

証明：由圖 2-6 與(2.35)式可知：

$$\because \left|LLR\left(q_{i'j}\right)\right| > 0 \;\Rightarrow\; \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right) > 0 \;\Rightarrow\; \sum_{i' \in L(j)\backslash\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right) > 0 \tag{2.41}$$

$$\therefore \Psi\left(\sum_{i'\in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)\right) > 0 \tag{2.42}$$

。亦可由(2.39)式中得知：

$$\because \left|LLR\left(q_{i'j}\right)\right| > 0 \;\Rightarrow\; \therefore \min_{i'\in L(j)\setminus\{i\}} \left|LLR\left(q_{i'j}\right)\right| > 0 \tag{2.43}$$

，因為$\Psi\left(\sum_{i'\in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)\right) > 0$，$\min_{i'\in L(j)\setminus\{i\}} \left|LLR\left(q_{i'j}\right)\right| > 0$，且兩者最後皆乘上$(-1)^{|L(j)|} \times \prod_{i'\in L(j)\setminus\{i\}} \operatorname{sgn}\left(LLR\left(q_{i'j}\right)\right)$之式，故$\operatorname{sgn}\left(LLR\left(r_{ij}\right)\right) = \operatorname{sgn}\left(LLR'\left(r_{ij}\right)\right)$。

特性二、$LLR\left(r_{ij}\right)$之絕對值恆小於$LLR'\left(r_{ij}\right)$之絕對值：

$$\left|LLR\left(r_{ij}\right)\right| < \left|LLR'\left(r_{ij}\right)\right| \tag{2.44}$$

証明：

$$\begin{aligned} \left|LLR\left(r_{ij}\right)\right| &= \Psi\left(\sum_{i'\in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)\right) \\ \left|LLR'\left(r_{ij}\right)\right| &= \min_{i'\in L(j)\setminus\{i\}} \left|LLR\left(q_{i'j}\right)\right| \end{aligned} \tag{2.45}$$

$$\sum_{i'\in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right) > \max_{i'\in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right) = \Psi\left(\min_{i'\in L(j)\setminus\{i\}} \left|LLR\left(q_{i'j}\right)\right|\right) \tag{2.46}$$

。由圖 2-6 可知：

$$|x_1| > |x_2| \;\Rightarrow\; \Psi\left(|x_1|\right) < \Psi\left(|x_2|\right) \tag{2.47}$$

，由(2.46)式之結果，可推得：

$$\begin{aligned} &\Psi\left(\sum_{i'\in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)\right) > \Psi\left(\Psi\left(\min_{i'\in L(j)\setminus\{i\}} \left|LLR\left(q_{i'j}\right)\right|\right)\right) \\ &\Rightarrow \Psi\left(\sum_{i'\in L(j)\setminus\{i\}} \Psi\left(\left|LLR\left(q_{i'j}\right)\right|\right)\right) < \min_{i'\in L(j)\setminus\{i\}} \left|LLR\left(q_{i'j}\right)\right| \\ &\Rightarrow \left|LLR\left(r_{ij}\right)\right| < \left|LLR'\left(r_{ij}\right)\right| \end{aligned} \tag{2.48}$$

。藉由此兩項特性，我們可對 $LLR'\left(r_{ij}\right)$ 乘上一個正規化參數“α”：

$$\begin{aligned} LLR''\left(r_{ij}\right) &= \alpha \times LLR'\left(r_{ij}\right) \\ &= \alpha \times \left(-1\right)^{\left|L(j)\right|} \times \prod_{i' \in L(j)\setminus\{i\}} \operatorname{sgn}\left(LLR\left(q_{i'j}\right)\right) \times \min_{i' \in L(j)\setminus\{i\}} \left|LLR\left(q_{i'j}\right)\right| \end{aligned} \tag{2.49}$$

，使 $LLR''\left(r_{ij}\right)$ 能比 $LLR'\left(r_{ij}\right)$ 更近似 $LLR\left(r_{ij}\right)$，而有較好的效能。而正規化參數“α”可由 $\left(2.50\right)$ 式求得：

$$\alpha = \frac{E\left\{\left|LLR\left(r_{ij}\right)\right|\right\}}{E\left\{\left|LLR'\left(r_{ij}\right)\right|\right\}} \tag{2.50}$$

2.4.3 正規化參數之推導

由於在 BPSK 和 AWGN 之狀態下所求出來的正規化參數，仍可適用於 MIMO-OFDM 系統與多重路徑通道（Multipath channel），故為了簡化運算我們假設系統使用 BPSK 調變與 AWGN 通道，並只考慮第一次遞迴所產生的效應，忽略在第二次遞迴後所造成的影響。若定義接收端所接收到的訊號為 y_R，雜訊機率分佈之變異數（variance）為 N_0，列權重為 ρ，則在第一次遞迴內，$LLR^{(1)}\left(q_{ij}\right)$ 會被初始化為 $\frac{4}{N_0} y_R$，我們可設定 $\left\{X_i : i = 1,2,\ldots,W\right\} = \left\{q_{i'j} : i' \in L\left(j\right)\setminus\left\{i\right\}\right\}$，其中 X_i 為 i.i.d.隨機變數，而 $W = \rho - 1$。

我們可由 $\left(2.22\right)$ 式、$\left(2.39\right)$ 式與上段所得之結果，得到：

$$E\left\{\left|LLR\left(r_{ij}\right)\right|\right\} = E\left\{\left|\log \frac{1-\left(-1\right)^{W}\prod_{i=1}^{W}\tanh\left(\frac{X_i}{2}\right)}{1+\left(-1\right)^{W}\prod_{i=1}^{W}\tanh\left(\frac{X_i}{2}\right)}\right|\right\} \tag{2.51}$$

$$E\left\{\left|LLR'\left(r_{ij}\right)\right|\right\} = E\left\{\min\left(\left|X_1\right|,\left|X_2\right|,\ldots\left|X_W\right|\right)\right\} \tag{2.52}$$

，並將結果代入 $\left(2.50\right)$，以計算正規化參數 α。

為了計算 $E\left\{\left|LLR'\left(r_{ij}\right)\right|\right\}$，令 $Y_i=\left|X_i\right|$，$i=1,2,\ldots W$，則 Y_i 的機率密度分佈函數（probability density function）為：

$$f_{Y_i}(y)=\left(f_{X_i}(y)+f_{X_i}(-y)\right)\times u(y)=2f_{X_i}(y)\times u(y) \tag{2.53}$$

，其中 $f_{X_i}(\bullet)$ 為 X_i 之機率密度分佈函數，$u(\bullet)$ 則為 unit-step 函式。因為 X_i 為 i.i.d. 隨機變數，所以 Y_i 亦為 i.i.d.隨機變數，故我們可由 (2.53) 式推得：

$$\begin{aligned} P\left(\left|LLR'\left(r_{ij}\right)\right|>y\right) &= P\left(\min\left(Y_1,Y_2,\ldots,Y_W\right)>y\right) \\ &= P\left\{Y_1>y,Y_2>y,\ldots Y_W>y\right\} \\ &= \left[P\left(Y_1>y\right)\right]^W \end{aligned} \tag{2.54}$$

，由於 $\left|LLR'\left(r_{ij}\right)\right|>0$，所以：

$$E\left\{\left|LLR'\left(r_{ij}\right)\right|\right\}=\int_0^\infty P\left(\left|LLR'\left(r_{ij}\right)\right|>y\right)dy \tag{2.55}$$

，接著我們將 (2.54) 式代入 (2.55) 式可得：

$$\begin{aligned} E\left\{\left|LLR'\left(r_{ij}\right)\right|\right\} &= \int_0^\infty \left[P\left(Y_1>y\right)\right]^W dy \\ &= \int_0^\infty \left[\int_y^\infty f_{Y_i}(y_1)\,dy_1\right]dy \\ &= \int_0^\mu \left[1-Q\left(\frac{\mu-y}{\sigma}\right)+Q\left(\frac{\mu+y}{\sigma}\right)\right]^W dy \\ &\quad +\int_\mu^\infty \left[Q\left(\frac{\mu-y}{\sigma}\right)+Q\left(\frac{\mu+y}{\sigma}\right)\right]^W dy \end{aligned} \tag{2.56}$$

，其中 $\mu=\frac{4}{N_0}$，$\sigma^2=\frac{8}{N_0}$，（$\because LLR^{(1)}\left(q_{ij}\right)=\frac{4}{N_0}y_R$），$Q(x)=\frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-\frac{x^2}{2}}dx$，

又因為 (2.56) 式之第二項之值極小，故可忽略，所以 (2.56) 式可再簡化為：

$$E\left\{\left|LLR'\left(r_{ij}\right)\right|\right\}\approx\int_0^\mu \left[1-Q\left(\frac{\mu-y}{\sigma}\right)+Q\left(\frac{\mu+y}{\sigma}\right)\right]^W dy \tag{2.57}$$

為了進一步推導(2.51)式，我們定義：

$$\beta = (-1)^W \prod_{i=1}^{W} \tanh\left(\frac{X_i}{2}\right) = \prod_{i=1}^{W} \frac{1-e^{X_i}}{1+e^{X_i}} \tag{2.58}$$

，則依據泰勒展開式（Taylor's series），可得到：

$$\log\frac{1-\beta}{1+\beta} = -2\left(\beta + \frac{\beta^3}{3} + \frac{\beta^5}{5} + \frac{\beta^7}{7} + \cdots\right) \tag{2.59}$$

，由於：

$$sign(\beta) = sign(\beta^3) = sign(\beta^5) = sign(\beta^7) = \cdots \tag{2.60}$$

，所以：

$$E\left\{\left|LLR(r_{ij})\right|\right\} = E\left\{\left|\log\frac{1-\beta}{1+\beta}\right|\right\} = 2\sum_{k=1}^{\infty}\frac{E\left\{|\beta|^{2k-1}\right\}}{2k-1} \tag{2.61}$$

，令 $m_k = E\left[|\beta|^k\right]$，則因為 X_i 為 i.i.d.隨機變數，故：

$$\begin{aligned} m_k &= E\left\{\left|(-1)^W \prod_{i=1}^{W}\tanh\left(\frac{X_i}{2}\right)\right|\right\} \\ &= E\left\{\left|\prod_{i=1}^{W}\tanh\left(\frac{X_i}{2}\right)\right|\right\} \\ &= E\left\{\left|\tanh\left(\frac{X_i}{2}\right)\right|\right\}^W \\ &= E\left\{\tanh\left(\frac{|X_i|}{2}\right)\right\}^W \\ &= E\left\{\tanh\left(\frac{Y_i}{2}\right)\right\}^W \end{aligned} \tag{2.62}$$

，由(2.62)式之結果，可求得：

$$E\left\{\left|LLR(r_{ij})\right|\right\} = 2\left(m_k + \frac{{m_k}^3}{3} + \frac{{m_k}^5}{5} + \frac{{m_k}^7}{7} + \cdots\right) \tag{2.63}$$

，最後，我們將(2.57)式與(2.63)式代入(2.50)式中，即可求得正規化參數 α。

由上述推導的關係式中，我們可發現，正規化參數α之值的大小，會與當時的 SNR（訊雜比，signal to noise ratio）之值有關，但實際上，由於正規化參數α對 SNR 的靈敏度（sensitive）並不高，我們只需尋找使位元錯誤率（bit error rate）介於10^{-3}到10^{-4}之間的某個 SNR 之值，並用此 SNR 之值來求得正規化參數α，此正規化參數α即可操作在任何 SNR 之值上。

2.4.4 Normalized BP-based 演算法解碼流程

第一步：初始化，設定最大遞迴次數“k_{Max}”，並令一開始的 check node 至 bit node 之“1”與“0”的機率資訊是相等的：

$$LLR''^{(0)}\left(r_{ij}\right)=\log\frac{0.5}{0.5}=0 \tag{2.64}$$

第二步：計算 bit node 至 check node 之機率資訊：

$$LLR^{(k)}\left(q_{ij}\right)=LLR\left(p_i\right)+\sum_{j'\in M(i)\backslash\{j\}}LLR''^{(k-1)}\left(r_{ij'}\right) \tag{2.65}$$

第三步：計算 check node 至 bit node 之機率資訊：

$$LLR''^{(k)}\left(r_{ij}\right)=\alpha\left(-1\right)^{|L(j)|}\prod_{i'\in L(j)\backslash\{i\}}\operatorname{sgn}\left(LLR^{(k)}\left(q_{i'j}\right)\right)\times\min_{i'\in L(j)\backslash\{i\}}\left|LLR^{(k)}\left(q_{i'j}\right)\right| \tag{2.66}$$

第四步：計算 bit node B_i 事後機率資訊，並進行判決：

$$LLR^{(k)}\left(B_i\right)=LLR\left(p_i\right)+\sum_{j\in M(i)}LLR''^{(k)}\left(r_{ij}\right) \tag{2.67}$$

$$B_i^{(k)}=\begin{cases}1 & LLR^{(k)}\left(B_i\right)\geq 0\\ 0 & LLR^{(k)}\left(B_i\right)<0\end{cases} \tag{2.68}$$

第五步：重複遞迴，直到解出的碼字符合$H_p\vec{V}^T=0$，或$k=k_{Max}$。

Normalized BP-based 演算法解碼流程，除了在第三步：計算 check node 至 bit node 之機率資訊外，其它步驟大致上與 Sum-Product 演算法解碼流程相同。

2.5 Normalized APP-based 演算法

2.5.1 APP-based 演算法

觀察 Sum-product 演算法中，bit node 至 check node 機率資訊之計算式(2.8)：

$$LLR(q_{ij}) = LLR(p_i) + \sum_{j' \in M(i) \setminus \{j\}} LLR(r_{ij'})$$

與 bit node B_i 機率資訊之計算式(2.26)：

$$LLR(B_i) = LLR(p_i) + \sum_{j' \in M(i)} LLR(r_{ij'})$$

，我們可發現兩式之形式非常接近，只相差一個 check node 至 bit node 機率資訊 $LLR(r_{ij})$，所以只要同位檢查矩陣 H 之行權重（column weight（行之元素為“1”的個數））夠大，則 $LLR(q_{ij})$ 與 $LLR(B_i)$ 會非常近似。在此情況下，我們可將所有的 bit node 至 check node 機率資訊，以 bit node 機率資訊來近似，也就是說，我們可省略掉 Sum-Product 演算法解碼流程中的第二步。透過此一化簡，我們可大幅的降低解碼時的運算量，也由於省略了 bit node 至 check node 機率資訊的計算，在硬體設計上更可節省掉將近一半的記憶體空間。

2.5.2 Normalized APP-based 演算法

我們可將 APP 演算法與 Normalized BP-based 演算法結合，做進一步的簡化，也就是把 APP 演算法中 check node 至 bit node 機率資訊計算式以(2.49)式取代。由於我們在計算正規化參數 α 時，只考慮第一次遞迴所產生的效應，忽略了第二次遞迴後所造成的影響，而在第一次遞迴內，APP 演算法的 $LLR(B_i)$（bit node B_i 之機率資訊）與 BP-based 演算法的 $LLR(q_{ij})$（bit node B_i 至 check node C_j

機率資訊）相同，皆會被初始化為 $LLR(p_i)$（another node N_i 至 bit node B_i 機率資訊），即：

$$LLR^{(0)}(B_i)_{APP-base} = LLR^{(1)}(q_{ij})_{BP-base} = LLR(p_i) \tag{2.69}$$

，故 Normalized APP-based 演算法與 Normalized BP-based 演算法之正規化參數 α 皆可用相同的計算式求得。而 Normalized APP-based 演算法之正規化參數 α 也只需尋找使位元錯誤率介於 10^{-3} 到 10^{-4} 之間的某個 SNR 之值，求得正規化參數 α 。

2.5.3 Normalized APP-based 演算法解碼流程

第一步：初始化，設定最大遞迴次數“k_{Max}”，並令一開始的 bit node B_i 之機率資訊為 another node N_i 至 bit node B_i 之機率資訊：

$$LLR^{(0)}(B_i) = LLR(p_i) \tag{2.70}$$

第二步：計算 check node 至 bit node 之機率資訊：

$$LLR''^{(k)}(r_{ij}) = \alpha(-1)^{|L(j)|} \prod_{i' \in L(j)\setminus\{i\}} \operatorname{sgn}\left(LLR^{(k-1)}(B_{i'})\right) \times \min_{i' \in L(j)\setminus\{i\}} \left|LLR^{(k-1)}(B_{i'})\right| \tag{2.71}$$

第三步：計算 bit node B_i 事後機率資訊，並進行判決：

$$LLR^{(k)}(B_i) = LLR(p_i) + \sum_{j \in M(i)} LLR''^{(k)}(r_{ij}) \tag{2.72}$$

$$B_i^{(k)} = \begin{cases} 1 & LLR^{(k)}(B_i) \geq 0 \\ 0 & LLR^{(k)}(B_i) < 0 \end{cases} \tag{2.73}$$

第四步：重複遞迴，直到解出的碼字符合 $H_p \vec{V}^T = 0$ ，或 $k = k_{Max}$ 。

在 Normalized APP-based 演算法解碼流程中，我們並不需要初始化 check node 至 bit node 之機率資訊，取而代之的，是對 bit node 之機率資訊進行初始化的動作。而最大的不同點，則是完全省略掉了 bit node 至 check node 機率資訊的計算，使得 Normalized APP-based 演算法解碼流程只有四個步驟，這亦是 Normalized APP-based 演算法能大幅降低運算量的原因。

2.6 Layered BP 演算法

2.6.1 Layered BP 演算法

整個 Layered BP 演算法的概念，就是將同位檢查矩陣 H_p 從水平（horizontal）方向分割成數個不同的層（layer），每一層稱為一個子矩陣（sub-matrix），並將每一個子矩陣視作一個單位的同位檢查矩陣來處理，進行解碼的運算。在同一次遞迴內，處理完一個子矩陣的解碼，若其解出的碼字符合 $H_p\bar{V}^T=0$，則停止；如果沒有，才再進行下一層的運算。一直到所有的子矩陣皆完成解碼的運算後，才算完成一次遞迴。

而其分割的原則：在同一個子矩陣內的任兩列，其為“1”的元素之位置不得重疊（non-overlaping），即其為“1”的元素之位置不得在同一行上，換言之，一子矩陣其行權重之值不能大於1。依據上述之特性，我們可發現；在一個子矩陣內，一個 check node 仍會與多個 bit node 連結，但一個 bit node 就只會連結到一個 check node，故我們必須對 Sum-product 演算法做部分的修正，才能對一個子矩陣進行解碼的運算。在此，我們先對單一個子矩陣做解碼的分析：

計算 bit node 至 check node 之機率資訊：

$$LLR^{(k)}\left(q_{ij}\right)=LLR^{(k)}\left(B_i\right)+LLR''^{(k-1)}\left(r_{ij}\right) \tag{2.74}$$

計算 check node 至 bit node 之機率資訊：

$$LLR''^{(k-1)}\left(r_{ij}\right)=2\times(-1)^{|L(j)|}\times\tanh^{-1}\left(\prod_{i'\in L(j)\setminus\{i\}}\tanh\left(\frac{1}{2}LLR^{(k)}\left(q_{ij}\right)\right)\right) \tag{2.75}$$

計算 bit node B_i 事後機率資訊：

$$LLR^{(k)}\left(B_i\right)=LLR^{(k)}\left(q_{ij}\right)+LLR''^{(k)}\left(r_{ij}\right) \tag{2.76}$$

。其中 (2.76) 式中的 $LLR\left(r_{ij}\right)$ 是由 (2.75) 式計算更新（update）所得之值。

觀察(2.74)式與(2.75)式，我們可發現，在(2.74)式中所更新的$LLR(q_{ij})$（bit node B_i 至 check node C_j 機率資訊），會立即被使用於$LLR(r_{ij})$（check node C_j 至 bit node B_i 機率資訊）的計算上，而 Layered BP 演算法也因為此項特性，可以只用較少的遞迴次數，便達到和 Sum-product 演算法相同的位元錯誤率；另外，因為我們是將一個子矩陣視為一個同位檢查矩陣來處理，而一個子矩陣的解碼運算中，最多只會有和碼字長度一樣多的 bit node 至 check node 之機率資訊要儲存，所以在此一架構下，硬體設計可節省掉將近一半的記憶體空間；這是 Layered BP 演算法的兩個優點。但 Layered BP 演算法仍有其缺點存在，其中一個是(2.75)式（check node 至 bit node 之機率資訊計算式），此計算式中的 tanh 及 $\tanh^{-1}$ 函式太過複雜，並不適合硬體設計；另一個則是其將同位檢查矩陣 H_p 分層處理的架構，需處理完一個子矩陣，才能進行下一個子矩陣的運算，而原本 Sum-product 演算法在解碼時，是以平行化運算的結構來進行解碼運算，但 Layered BP 演算法分層處理的架構卻破壞了其平行化處理的優點，降低了解碼的速度。

2.6.2 Layered BP 演算法修正

為了解決上述的兩個缺點，吾人提出兩個方法，對 Layered BP 演算法作些微的修正，以處理這兩個缺點。其中第一個缺點，我們可將 Layered BP 演算法與 Normalized BP-based 演算法結合（可稱之為 Layered Normalized BP-based 演算法），把(2.75)式以(2.49)式取代。同樣地，我們在計算正規化參數α時，也只考慮在第一次遞迴內第一個子矩陣所造成的影響，而不去考慮第二個子矩陣之後的效應，而在第一次遞迴的第一個子矩陣內，Layered BP 演算法的$LLR(q_{ij})$（bit node B_i 至 check node C_j 機率資訊），也一樣與 BP-based 演算法的$LLR(q_{ij})$相同，

皆會被初始化為 $LLR(p_i)$（another node N_i 至 bit node B_i 機率資訊），即：

$$LLR^{(1)}\left(q_{ij}\right)_{Layer\_BP} = LLR^{(1)}\left(q_{ij}\right)_{BP-base} = LLR\left(p_i\right) \tag{2.77}$$

，故 Layered Normalized BP-based 之正規化參數 α 之值與 Normalized BP-based 演算法之正規化參數 α 之值相同，可用相同的計算式求得。

而第二個缺點，由於將同位檢查矩陣 H_p 任意兩列向量進行置換，並不會影響到碼字的結構，亦不會改變解碼器的架構，仍滿足 $H_p\vec{V}^T = 0$ 的關係式。故我們可透過將同位檢查矩陣 H_p 的列向量做交換的運作，來增加其可平行化處理的部份。原來的 Layered BP 演算法對同位檢查矩陣 H_p 進行子矩陣的分割時，某一子矩陣“A”之下一層子矩陣“B”的第一個列“b_1”，必然與子矩陣 A 的某列，其為“1”的元素之位置有重疊；但在子矩陣 B 的列 b_1 之後的所有列（含下一層之後之所有子矩陣的所有列），卻有可能符合「其為“1”的元素之位置不得重疊」之限制，也就是說這些能符合此限制的列，皆可於子矩陣 A 同屬於同一個子矩陣。所以我們可藉由列交換的運作，讓同位檢查矩陣 H_p 分割出來的子矩陣數目較使用方法原本分割出來的子矩陣數目少，增加 Layered BP 演算法可平行化處理的部份。

我們就此一分割方法舉例說明：

若同位檢查矩陣：

$$H_p = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2.78}$$

，依照原本的方式進行分割，可分為 6 層：

$$H_p = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2.79}$$

，若使用修正後的方式進行分割，則會分為3層：

$$H_p = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2.80}$$

2.6.3 Layered Normalized BP-based 演算法解碼流程

若同位檢查矩陣 H 被分割成 M 層子矩陣：

第一步：初始化，設定最大遞迴次數“k_{Max}”，並令一開始的 check node 至 bit node 之“1”與“0”的機率資訊是相等的：

$$LLR''^{(0)}\left(r_{ij}\right) = \log\frac{0.5}{0.5} = 0 \tag{2.81}$$

，而 bit node B_i 之機率資訊則為 another node N_i 至 bit node B_i 之機率資訊：

$$LLR^{(0,1)}\left(B_i\right) = LLR\left(p_i\right) \tag{2.82}$$

，其中，上標“(k,m)”內的數字“k”，表示為第 k 次遞迴所得之結果，而數字“m”，

則表示為第 m 層子矩陣內之計算。

第二步：計算 bit node 至 check node 之機率資訊：

$$LLR^{(k)}\left(q_{ij}\right)=LLR^{(k,m)}\left(B_i\right)+LLR''^{(k-1)}\left(r_{ij}\right) \tag{2.83}$$

第三步：計算 check node 至 bit node 之機率資訊：

$$LLR''^{(k)}\left(r_{ij}\right)=\alpha\left(-1\right)^{|L(j)|}\prod_{i'\in L(j)\setminus\{i\}}\operatorname{sgn}\left(LLR^{(k)}\left(q_{i'j}\right)\right)\times\min_{i'\in L(j)\setminus\{i\}}\left|LLR^{(k)}\left(q_{i'j}\right)\right| \tag{2.84}$$

第四步：計算 bit node B_i 事後機率資訊，並進行判決：

$$LLR^{(k,m+1)}\left(B_i\right)=LLR^{(k)}\left(q_{ij}\right)+LLR''^{(k)}\left(r_{ij}\right) \tag{2.85}$$

$$B_i^{(k)}=\begin{cases}1 & LLR^{(k)}\left(B_i\right)\geq 0\\ 0 & LLR^{(k)}\left(B_i\right)<0\end{cases} \tag{2.86}$$

第五步：若解出的碼字符合 $H_p\bar{V}^T=0$，停止解碼之運算；若不是，則移往下一層子矩陣，重複遞迴，直到解出的碼字符合 $H_p\bar{V}^T=0$，或 $\left(k=k_{Max}\ \&\ m=M\right)$。

2.7 模擬結果

我們將上述所提到的 LDPC Codes 解碼演算法，使用 $(273,191)$ Gallager Codes，在 BPSK 系統、AWGN 通道環境下進行測試，並將這些 LDPC Codes 之解碼演算法的效能和 Sum-product 演算法的效能做比較。

由圖 2-7 與圖 2-8 之模擬結果，可知經由正規化參數修正後的演算法，有較低的位元錯誤率，其效能更接近 Sum-product 演算法的模擬結果，而其中的 df（dynamic factor，依照其所在之 SNR 來計算其正規化參數之值）與 cf（constant factor，正規化參數之值固定由介於10^{-3}到10^{-4}之間的某個 SNR 所求得）兩者之模擬結果機乎是相同的，由介於10^{-3}到10^{-4}之間的 SNR 之值所計算之正規化參數確實可操作在任何 SNR 上。

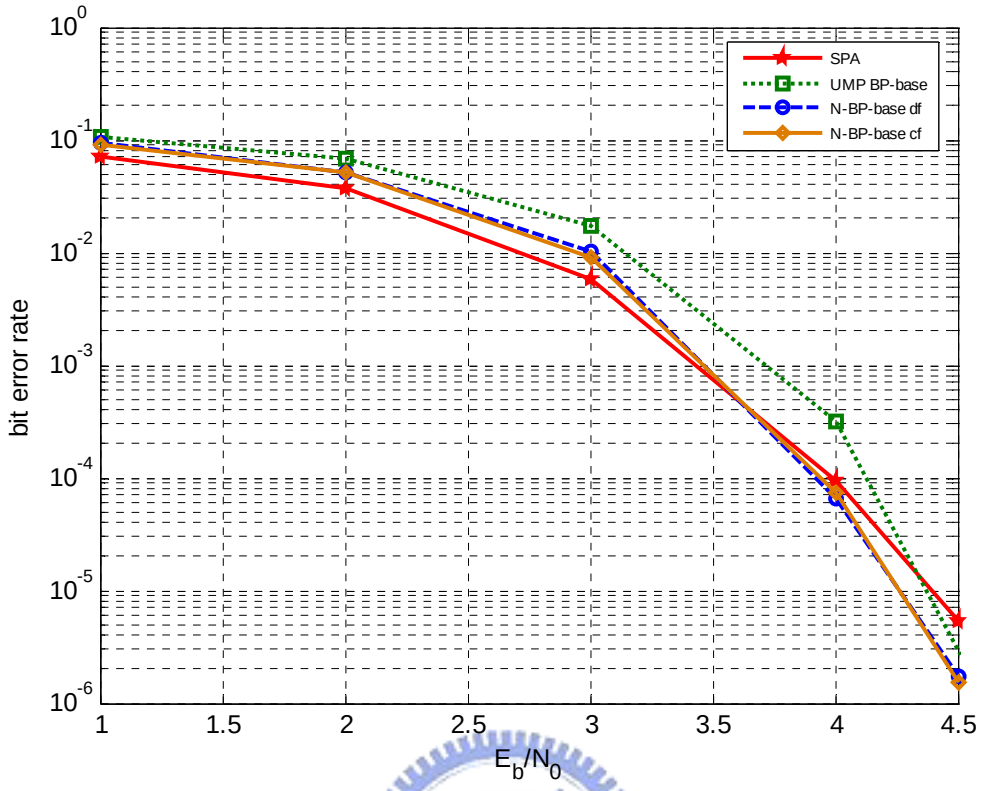


圖 2-7 Normalized BP-based 演算法模擬結果

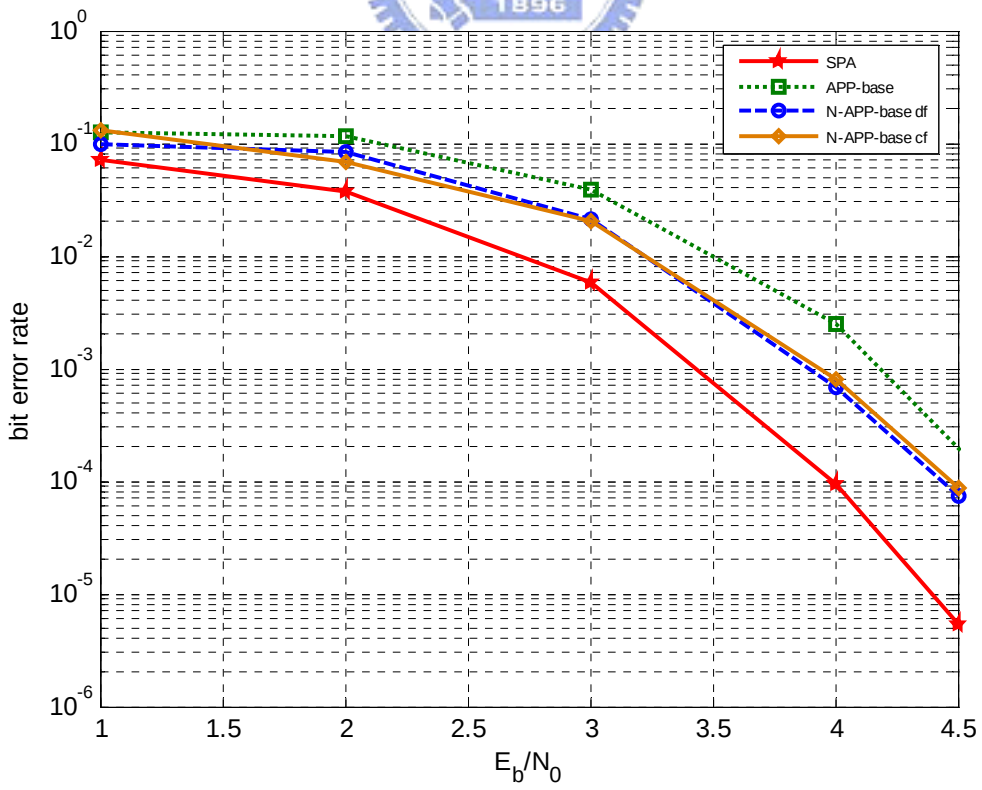


圖 2-8 Normalized APP-based 演算法模擬結果

在圖 2-9 之模擬結果中，L-N-BP-based unrc（non row change）表示使用未修正前之方式來對同位檢查矩陣進行子矩陣的分割，L-N-BP-based rc（row change）則是表示可執行列交換的運作來對同位檢查矩陣進行分割。由圖 2-9 可知，在最大遞迴次數為10的條件下，不論是 L-N-BP-based rc 或是 L-N-BP-based unrc，都比 Sum-product 演算法的位元錯誤率低，就算是最大遞迴次數為50的情況，L-N-BP-based rc 和 L-N-BP-based unrc 的模擬結果都近似於 Sum-product 演算法的模擬結果；另外，由圖 2-9 亦可發現，在同樣的遞迴次數下，L-N-BP-based rc 會比 L-N-BP-based unrc 有更好的效能。

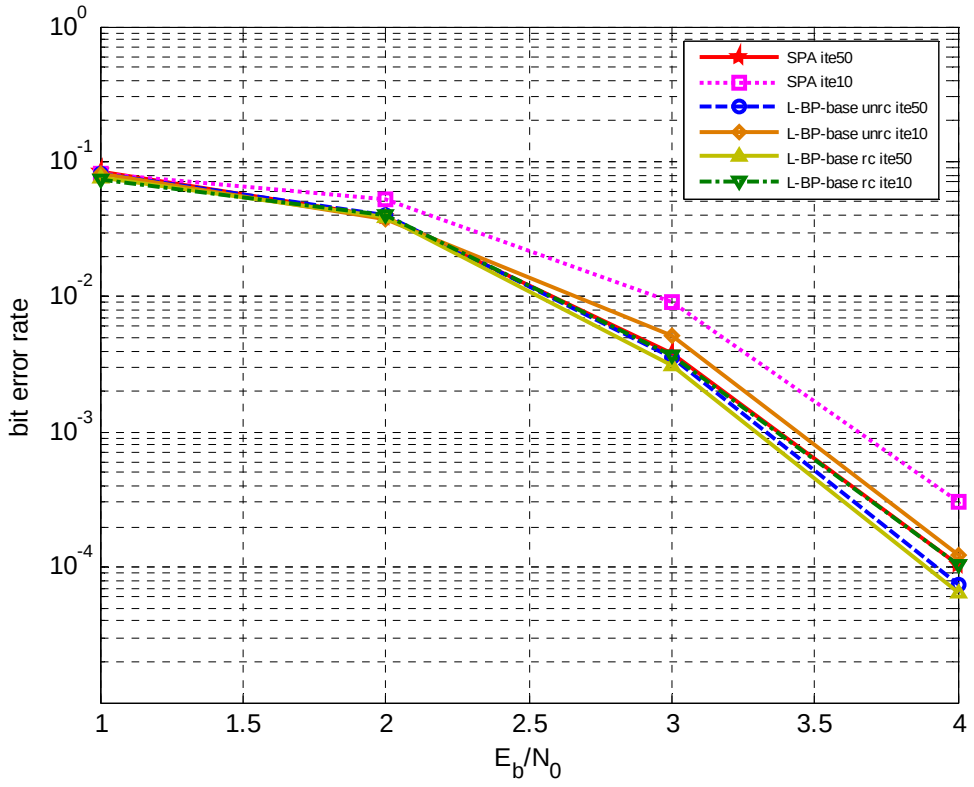


圖 2-9 Layered BP 演算法模擬結果

由圖 2-10 之模擬結果可知，即使用 Normalized BP-based 演算法中之(2.49)式來取代 Layered BP 演算法之(2.75)式，其效能與位元錯誤率，仍與 Sum-product 演算法之結果非常接近，其效能和原來的 Layered BP 演算法相比，只有輕微的下降（degression）。

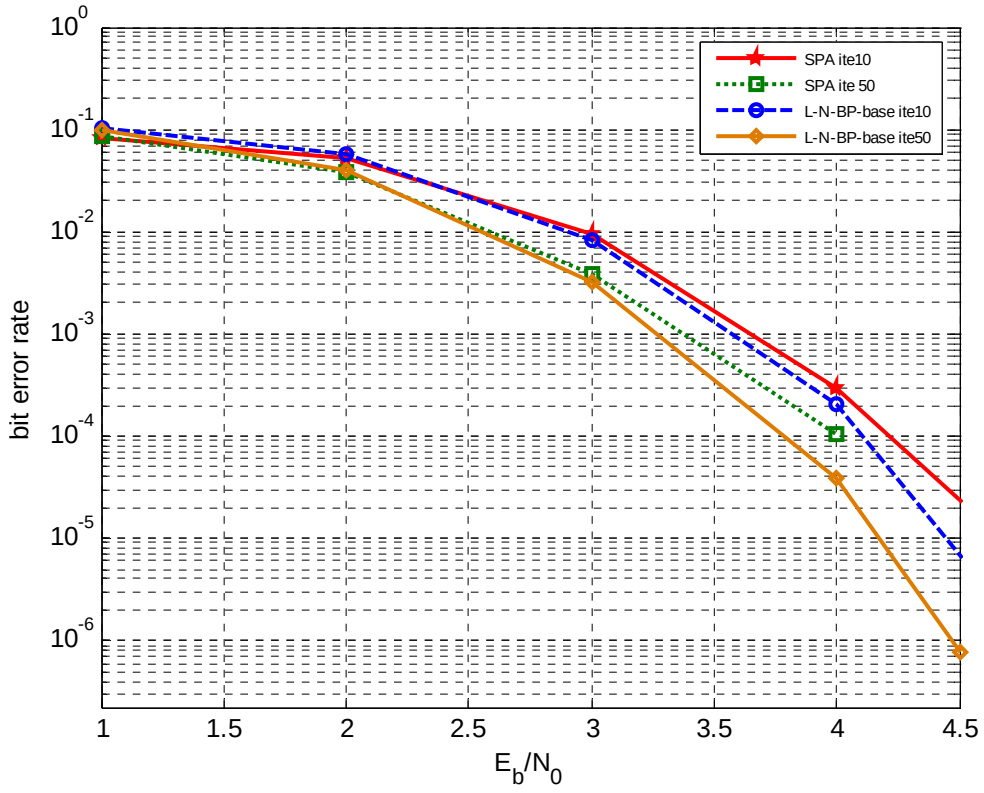


圖 2-10 Layered Normalized BP-based 演算法模擬結果

第3章 MIMO-OFDM 系統

3.1 MIMO-OFDM 系統

我們考慮一個使用 LDPC Codes 的 MIMO-OFDM 的系統，其中包括了 N_t 個傳送天線及 N_r 個接收天線，系統如圖 3-1 所示。傳送端包括了 LDPC Codes 編碼器，將位元對映到 $\left(2^M\right)$ QAM 訊號的 Mapping 及 OFDM 系統中 IFFT（size N_c）及循環前序（cyclic prefix）的插入。接收端則主要由 OFDM-demodulation 的 FFT 及 CP 移除、符元的反對映（demapping）及 LDPC Codes 解碼器所構成。這邊要特別說明的是，由於 LDPC Codes 的同位檢查矩陣有稀疏的特性，使得其在編碼的時候不需要交錯器（interleaver），因為位元的資訊會相隔的比較遠，與交錯器有著同樣的效果。

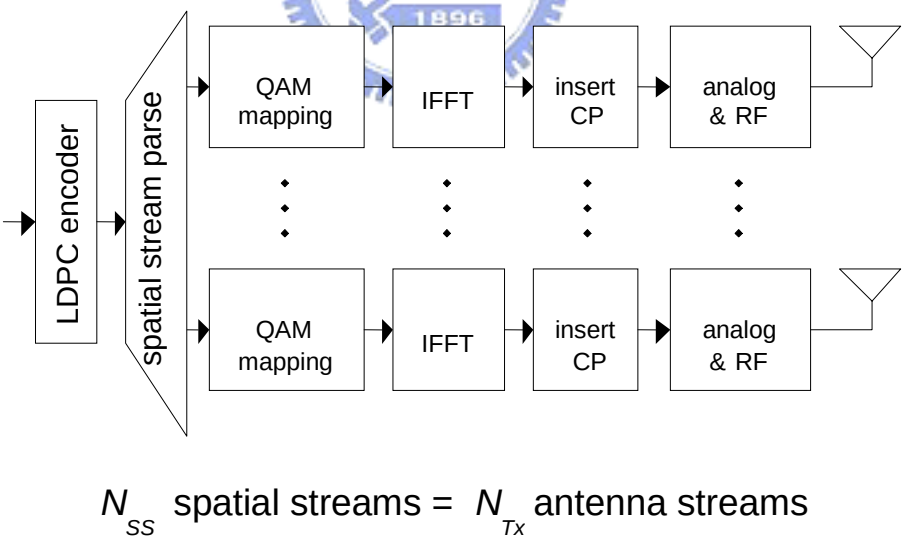


圖 3-1 MIMO-OFDM 系統傳送端方塊圖

資料位元一開始經過一個碼率（code rate）為 R_c 的 LDPC Codes 編碼器，這些經過編碼的位元資料接著會被以大小為 $N_t \times M \times N_c$ 分塊，成為多天線的 OFDM

symbol。在這塊的資料中又以大小為 $N_t \times M$ 分群，對應到 $N_t \times 1$ 複數訊號的向量，令此向量為 $S(k,n)$，$k = 1,2,...N_c$，表示在第 n 個 OFDM symbol 中的第 k 個子載波中的訊號。之後對應至相同 OFDM symbol 的訊號便通過 IFFT 及加 CP 方塊。每根多天線的 OFDM symbol 的資料率為 $R_c \times N_t \times M \times N_c$。

我們假設 MIMO 通道為頻率選擇衰落通道（frequency selective channel），並有適當長度的 CP 及完美的同步。在頻域上，在第 n 個 OFDM symbol 的第 k 個子載波（subcarrier）中，假設傳送訊號為 $X(k,n)$，我們將會收到 $N_r \times 1$ 的複數訊號 $Y(k,n)$ $k = 1,2,...N_c$。

$$Y(k,n) = H(k,n)S(k,n) + N(k,n) \tag{3.1}$$

，其中 $N(k,n)$ 是指 AWGN 而 $H(k,n)$ 則是指在第 n 個 OFDM symbol 中的第 k 個子載波中的通道。$H(k,n)$ 的第 (i,j) 項則為從第 i 根傳送天線到第 j 個接收天線的增益（gain）。另外在經過 LDPC Codes 編碼器後，我們可假設 $H(k,n)$ 與 $N(k,n)$ 對不同的 k 和 n 而言是 i.i.d.隨機變數。而為了表示方便，在本章的數學模式中都將 n 省略，也就是 (3.1) 式可表示為 (3.2) 式：

$$Y(k) = H(k)S(k) + N(k) \tag{3.2}$$

3.2 MIMO MMSE 偵測

在上一節中，我們可以將MIMO-OFDM系統如 (3.2) 所示，傳送訊號 $S(k)$ 經過通道 $H(k)$ 後，加上高斯雜訊 $N(k)$ 得到接收訊號 $Y(k)$。在空間多工的應用中，MIMO偵測（detection）就是要在接收端將其它根天線的訊號消除，從 $Y(k)$ 去推算出 $S(k)$，在本論文中所使用的是最小均方差估測（MMSE）[13]和以最大可能

性解碼為基礎的List Sphere decoding（LSD，亦可稱為Sphere List detection（SLD））演算法 [22]，其中List Sphere decoding演算法將留待下一章討論。

假設收到接收訊號的條件下：$Y(k)=H(k)S(k)+N(k)$，我們定義一個錯誤向量$e(k)$，此向量代表傳送的訊號與接收訊號乘上 MMSE 壓抑矩陣後的誤差，也就是$e(k)=Y(k)-G^{H}Y(k)$，其中G是 MMSE 壓抑矩陣。另外，定義一個成本函數（Cost Function）J，如下式：

$$J=E\left\{e^{H}(k)e(k)\right\}=tr\left[E\left\{e(k)e(k)^{H}\right\}\right] \tag{3.3}$$

，為了使成本最小，我們對J微分，使其等於零，並找出此時的 MMSE 壓抑矩陣的值，於是可以推導出 Wiener-Hopf equation：

$$G^{H}R_{YY}=R_{SY} \tag{3.4}$$

，其中

$$\begin{aligned} R_{YY}&=E\left\{Y(k)Y^{H}(k)\right\}\\ R_{XY}&=E\left\{S(k)Y^{H}(k)\right\}\end{aligned} \tag{3.5}$$

，前者為接收訊號向量的協方差矩陣（Covariance Matrix），後者為傳輸和接收向量的交互相關矩陣（Cross-Correlation Matrix）。然後我們假設: $E\left[SS^{H}\right]=\frac{1}{\alpha}I_{Nt}$，$E\left[NN^{H}\right]=I_{Nr}$，$E\left[SN^{H}\right]=0$，這裡$\alpha=\frac{N_t}{\sigma^2}$。換句話說，訊號$S$及外加雜訊$N$在空間上是白色（white）及沒有相關的隨機向量，而變異數分別為$\frac{1}{\alpha}I_{Nt}$及I_{Nr}（我們可以將$\frac{1}{\alpha}$想像成在每根天線上 SNR），便可以推導出 MMSE 壓抑矩陣G如下：

$$G=\left[HH^{H}+\alpha I_{N_r\times N_r}\right]^{-1}H \tag{3.6}$$

，所估測的訊號則可表示為：

$$\hat{S}(k)=G\cdot Y(k) \tag{3.7}$$

3.3 軟性反對映（Soft demapping）

符元對映（symbol mapping）在 M-ary 的系統中扮演著很重要的角色，在非遞迴解碼裡，証明了葛雷碼（Gray-code）對映的最佳性。然而，在遞迴解碼裡葛雷 mapping 便並沒有辦法提供明顯的效果，這是因為在相鄰的$\left(2^M\right)$QAM 裡，對映位元最多只相差 2 位元而已，如此一來遞迴迴授並沒有辦法明顯地改善 soft-bit metrics。在使用葛雷編碼下，如果使用最大可能性解碼時，則反對映（demapping）則有明顯簡化的方法，我們將在下面說明。

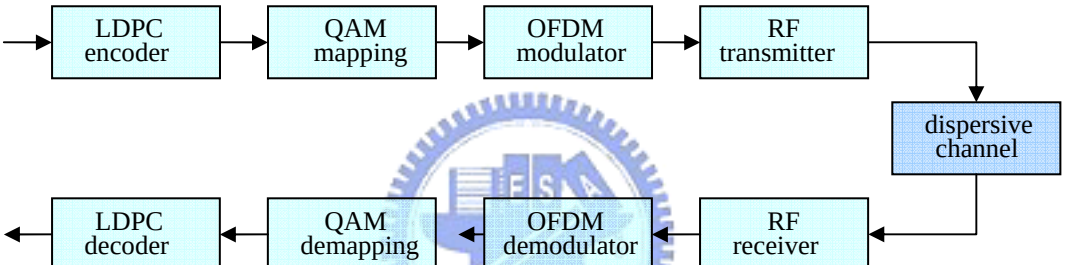


圖 3-2 單天線傳送接收器

所謂的軟性反對映（soft demapping）[14]即在決定在每個接收到的symbol裡的位元的機率是1或是0，metric的值愈大便代表著位元是1的機率愈大，反之值愈小的話便代表著位元是0的機率大。在說明應用在MIMO的情況之前，我們先說明在單天線的情況。

單天線傳送接收器如圖 2-2 所示，在頻域上訊號可以表示為：

$$Z(k)=H(k)S(k)+N(k) \tag{3.8}$$

，利用頻域等化器，我們可以有以下的結果：

$$Y(k)=\frac{Z(k)}{H(k)}=S(k)+\frac{N(k)}{H(k)}=S(k)+N'(k) \tag{3.9}$$

，$H(k)$ 是通道頻率響應（Channel Frequency Response，CFR）在第 k 個子載波的複數係數，S 是傳送訊號，N 則是指外加雜訊，$\sigma_{N'}^2 = \frac{\sigma_N^2}{\|H[k]\|^2}$。傳送訊號 S 是一個從有限星狀圖（constellation）$\Upsilon = \{S_1, S_2, ...S_{|\Upsilon|}\}$ 取出的 QAM symbol，由資料位元 $b = [b_0, b_1, ...b_M]$ 所對映而來。位元數 M 則是根據使用的 QAM 大小所決定。在接收到每個 $Z(n)$ 都有 $4M$ 個 metric 需要計算，in-phase 和 quadrature 位元 $b_{I,r}$、$b_{Q,r}$ 各需計算 0 和 1 的可能性，對 $b_{I,r}$ ($b_{Q,r}$ 是同樣的)而言，我們將 QAM 集合 Υ 分成二部分，$S_{I,r}^{(0)}$ 代表著在位置 (I,r) 的位元是 0 的點，也就是在 in-phase(I)位元群中第 r 個位元。相對的，是 1 的點便用 $S_{I,r}^{(1)}$ 表示。如此這二個 metrics 即可由下式表示：

$$m_c(b_{I,r}) = \max_{\alpha \in S_{I,r}^{(c)}} \log p(Z(k) | S(k) = \alpha), \quad c = 0,1 \tag{3.10}$$

，$Z(k)$ 的機率是個高斯函數：

$$p(Z(k) | S(k) = \alpha) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{ -\frac{1}{2} \frac{|Z(k) - H(k) \times \alpha|^2}{\sigma^2} \right\} \tag{3.11}$$

，且 $Z(n) = H(n) \times Y(n)$，如此 $(3,10)$ 即可改寫成：

$$m_c(b_{I,r}) = |H(k)|^2 \min_{\alpha \in S_{I,r}^{(c)}} |Y(k) - \alpha|^2, \quad c = 0,1 \tag{3.12}$$

，則依據 (2.7) 式 LLR 之定義：

$$\begin{aligned} L(b_{I,r}) &\equiv \log \frac{p(b_{I,r} = 1 | Z(k))}{p(b_{I,r} = 0 | Z(k))} \\ &= \log \frac{\sum_{\alpha \in S_{I,r}^{(1)}} p(S(k) = \alpha | Z(k))}{\sum_{\alpha \in S_{I,r}^{(0)}} p(S(k) = \alpha | Z(k))} \end{aligned} \tag{3.13}$$

提供了第 (I,r) 的位元為 1 或 0 之可靠度計算方法，由 $L(b_{I,r})$ 的正負號便可決定

in-phase(I)位元群中第r個位元為1或0，而值的大小則表示著可靠度。當雜訊並非很大時，我們可以將 log-sum 簡化成：

$$\log \sum_j Z_j \approx \max_j \log Z_j \tag{3.14}$$

，如此(3.13)式可寫成：

$$L\left(b_{I,r}\right) \approx \log \frac{\max\limits_{\alpha \in S_{I,r}^{(1)}} p\left(S(k)=\alpha \mid Z(k)\right)}{\max\limits_{\alpha \in S_{I,r}^{(0)}} p\left(S(k)=\alpha \mid Z(k)\right)} \tag{3.15}$$

，將(3.11)式代入(3.15)式可得：

$$L\left(b_{I,r}\right) = \frac{\left|H(k)\right|^2}{2\sigma^2}\{\min_{\alpha \in S_{I,r}^{(0)}} \left|Y(k)-\alpha\right|^2 - \min_{\alpha \in S_{I,r}^{(1)}} \left|Y(k)-\alpha\right|^2\} \tag{3.16}$$

在圖 3-3 我們以16QAM 為例：為了計算$L\left(b_{I,r}\right)$而在$\left(S_{I,r}^{(1)}, S_{I,r}^{(0)}\right)$這二個集合裡找離接收訊號最近的點時，我們可以發現在這二個集合裡最近的點都是會在同一條垂直的線上，同樣地為了計算$L\left(b_{I,r}\right)$在$\left(S_{I,r}^{(1)}, S_{I,r}^{(0)}\right)$裡找最近的點時，這二點便會在同一條水平線上。因此(3.16)式可以簡化成：

$$\begin{aligned} L\left(b_{I,r}\right) &= \frac{\left|H(k)\right|^2}{2\sigma^2}\left\{\min_{\alpha \in S'^{(0)}_{I,r}} \left|Y(k)-\alpha\right|^2 - \min_{\alpha \in S'^{(1)}_{I,r}} \left|Y(k)-\alpha\right|^2\right\} \\ &= \frac{\left|H(k)\right|^2}{2\sigma^2} \times 4 \times \frac{1}{4} \times \left\{\min_{\alpha \in S'^{(0)}_{I,r}} \left|Y(k)-\alpha\right|^2 - \min_{\alpha \in S'^{(1)}_{I,r}} \left|Y(k)-\alpha\right|^2\right\} \\ &= \frac{2\left|H(k)\right|^2}{\sigma^2} \times \frac{1}{4}\left\{\min_{\alpha \in S'^{(0)}_{I,r}} \left|Y(k)-\alpha\right|^2 - \min_{\alpha \in S'^{(1)}_{I,r}} \left|Y(k)-\alpha\right|^2\right\} \\ &\equiv CSI \times D_{I,r} \end{aligned} \tag{3.17}$$

，式中子集合$S'^{(c)}_{I,r}$定義為：$S'^{(c)}_{I,r} \equiv \Re\left\{S^{(c)}_{I,r}\right\}$，$c=0,1$。因此在計算$L\left(b_{Q,r}\right)$便是先由通道和雜訊算出在(3.17)式中所需的 CSI（channel state information），而$D_{I,r}$可利用圖 3-3 而計算得(3.18)式：

$$D_{I,1}=\begin{cases}\frac{1}{4}\left[\left(Y_I(k)+1\right)^2-\left(Y_I(k)-1\right)^2\right]=Y_I(k), & |Y_I(k)|\le 2\\ \frac{1}{4}\left[\left(Y_I(k)+1\right)^2-\left(Y_I(k)-3\right)^2\right]=2\left(Y_I(k)-1\right), & Y_I(k)<2\\ \frac{1}{4}\left[\left(Y_I(k)+3\right)^2-\left(Y_I(k)-1\right)^2\right]=2\left(Y_I(k)+1\right), & Y_I(k)>-2\end{cases} \quad (3.18)$$

$$D_{I,2}=\begin{cases}\frac{1}{4}\left[\left(Y_I(k)-1\right)^2-\left(Y_I(k)-3\right)^2\right]=-Y_I(k)+2, & Y_I(k)>0\\ \frac{1}{4}\left[\left(Y_I(k)+3\right)^2-\left(Y_I(k)-1\right)^2\right]=Y_I(k)+2, & Y_I(k)<0\end{cases}$$

，可將(3.18)式推導為：

$$D_{I,1}=\begin{cases}Y_I(k), & \left|Y_I(k)\right|\le 2\\ 2\left(Y_I(k)-1\right), & Y_I(k)>2\\ 2\left(Y_I(k)+1\right), & Y_I(k)<-2\end{cases} \quad (3.19)$$

$$D_{I,2}=-\left|Y_I(k)\right|+2$$

同理可得$D_{Q,r}$之式子，只要將(3.19)式中的$Y_I(k)$改成$Y_Q(k)$即可。

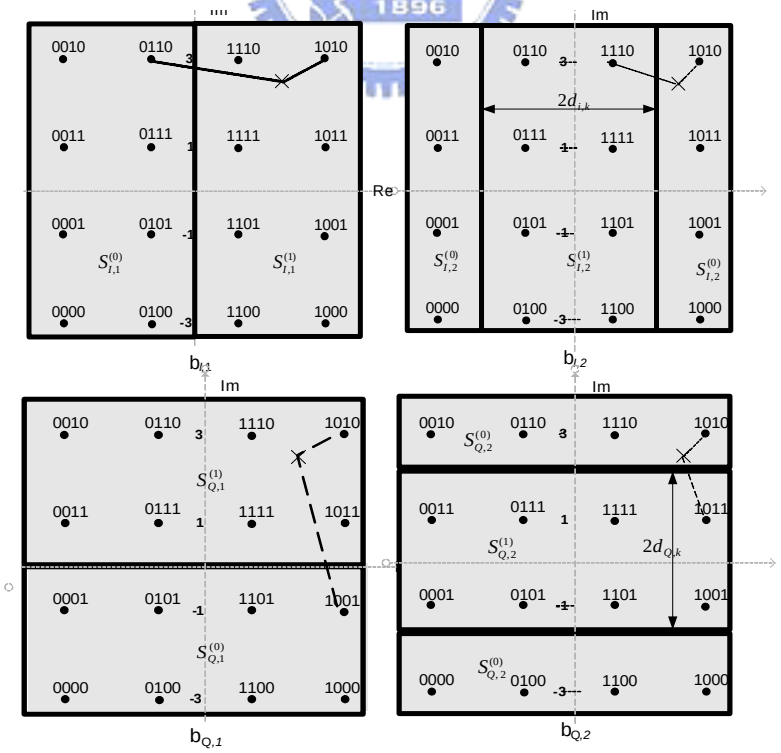


圖 3-3 16QAM 星狀圖的分割示意圖

(3.19)式在計算上仍略嫌複雜，但只要犧牲少許效能，則可以再進一步化簡為：

$$\begin{aligned} D_{I,1} &\cong Y_I(k) \\ D_{I,2} &= -\left|Y_I(k)\right| + 2 \end{aligned} \tag{3.20}$$

由(3.19)式與(3.20)式所算出之$D_{I,1}$及$D_{I,2}$的比較如圖 3-4 所示：

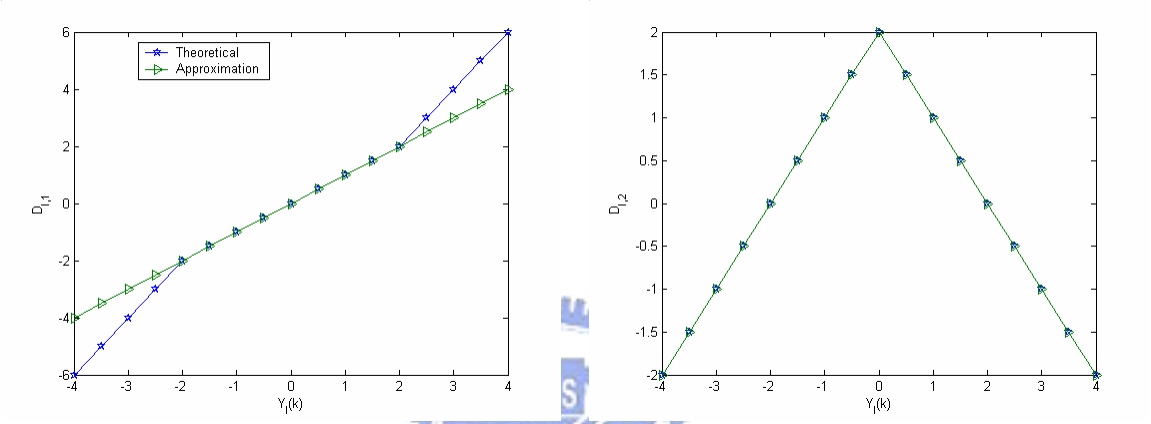


圖 3-4 In-phase 位元中，16QAM 之簡化對精確 LLR 計算方法比較圖

同樣地，在 64QAM 的情況下可以導出(3.21)式及其簡化式(3.22)式：

$$D_{I,1} = \begin{cases} Y_I(k), & \left|Y_I(k)\right| \le 2 \\ 2\left(Y_I(k)-1\right), & 2 < Y_I(k) \le 4 \\ 3\left(Y_I(k)-2\right), & 4 < Y_I(k) \le 6 \\ 4\left(Y_I(k)-3\right), & Y_I(k) > 6 \\ 2\left(Y_I(k)+1\right), & -4 \le Y_I(k) < -2 \\ 3\left(Y_I(k)+2\right), & -6 \le Y_I(k) < -4 \\ 4\left(Y_I(k)+3\right), & Y_I(k) < -6 \end{cases}$$

$$D_{I,2} = \begin{cases} 2\left(-\left|Y_I(k)\right|+3\right), & \left|Y_I(k)\right| \le 2 \\ 4-\left|Y_I(k)\right|, & 2 < \left|Y_I(k)\right| \le 6 \\ 2\left(-\left|Y_I(k)\right|+5\right) & \left|Y_I(k)\right| > 6 \end{cases} \tag{3.21}$$

$$D_{I,3} = \begin{cases} \left|Y_I(k)\right|-2, & \left|Y_I(k)\right| \le 4 \\ -\left|Y_I(k)\right|+6, & \left|Y_I(k)\right| > 4 \end{cases}$$

$$
\begin{aligned}
D_{I,1} &\cong Y_I(k) \\
D_{I,2} &\cong -\left|Y_I(k)\right| + 4 \\
D_{I,3} &\cong -\left|-\left|Y_I(k)\right| + 4\right| + 2
\end{aligned}
\tag{3.22}
$$

(3.21)式及(3.22)式所算出之$D_{I,1}$、$D_{I,2}$及$D_{I,3}$的比較如圖 3-5 所示：

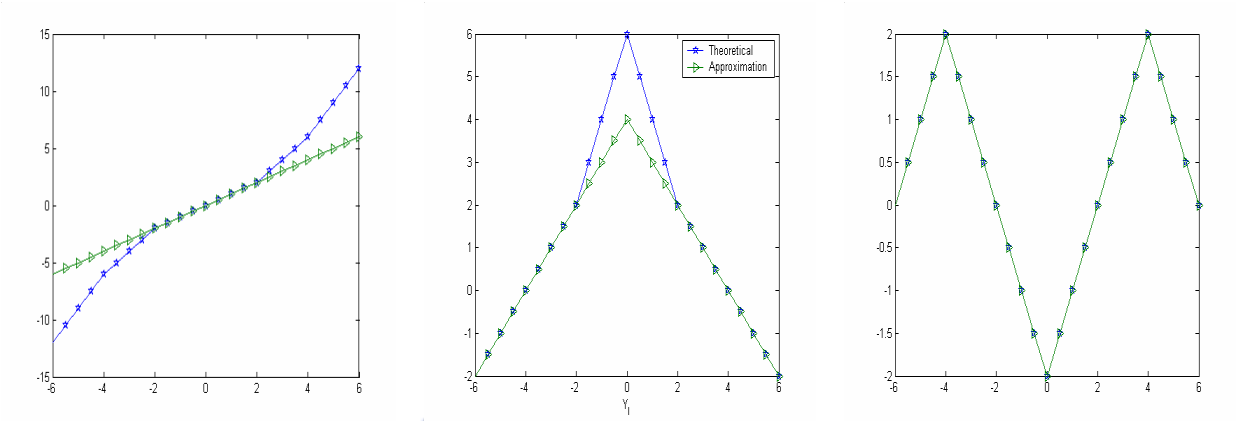


圖 3-5 In-phase 位元中，64QAM 之簡化對精確 LLR 計算方法比較圖

(3.22)式可以推展到使用類似葛雷編碼的任意平方 QAM 的星狀圖上，假設$d_{I,r}$及$d_{Q,r}$為當$r>1$時，二個集合之間距離的一半當如圖 3-3 所示，則從(3.19)式到(3.20)式便可一般化為：

$$
\begin{aligned}
D_{I,r} &= \begin{cases} Y_I(k), & r=1 \\ -\left|D_{I,r-1}\right| + d_{I,r}, & r>1 \end{cases} \\
L\left(b_{I,r}\right) &= CSI \times D_{I,r}
\end{aligned}
\tag{3.23}
$$

$$
\begin{aligned}
D_{Q,r} &= \begin{cases} Y_Q(k), & r=1 \\ -\left|D_{Q,r-1}\right| + d_{Q,r}, & r>1 \end{cases} \\
L\left(b_{Q,r}\right) &= CSI \times D_{Q,r}
\end{aligned}
$$

3.4 軟性輸入軟性輸出之 MIMO MMSE 接收機

這個接收器如圖 3-6 所示，為了方便說明，這裡用 2×2 的模型，不過這個接收器也很容易推廣到多根傳送接收天線模型。相對於將接收訊號視為訊號向量，我們直接使用線性的接收器將此向量分成二條獨立的資料流，並分開地計算其軟性輸出（Soft output）。

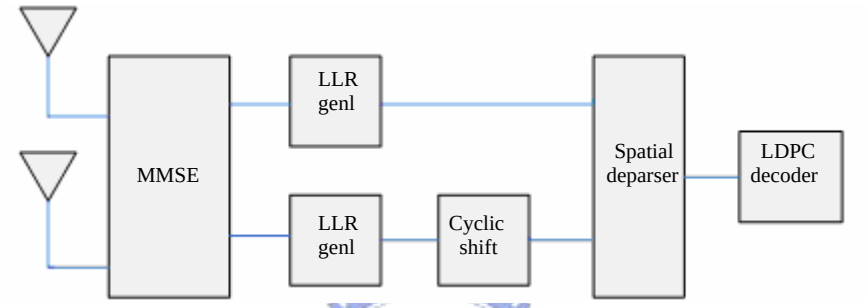


圖 3-6 2x2 之軟性輸入軟性輸出 MMSE 接收器

在 2×2 的情況下 (3.2) 式中的接收訊號可以表示為：

$$Y_k = \begin{bmatrix} h_{1,k} & h_{2,k} \end{bmatrix} \begin{bmatrix} s_{1,k} \\ s_{2,k} \end{bmatrix} + N_k \tag{3.24}$$

，$h_{i,k}$ 表示 $s_{i,k}$ 所受到的通道影響，也就是 H 的第 i 列。而 MMSE 的等化器則可以表示為：

$$W = \left(H_k^H H_k + \frac{2}{\rho} I \right)^{-1} H_k^H = \begin{bmatrix} w_1^T \\ w_2^T \end{bmatrix} \tag{3.25}$$

ρ 是 SNR $\frac{E_S}{N_0}$,則此等化過後的訊號即變成 $Z_k = WY_k$，假設 w_i^T 是 W 的第 i 列，則對第 i 根天線而言，等化後的訊號即可表示為：

$$Z_{i,k} = \underbrace{w_i^T h_{i,k}}_{H_{eff}} s_{i,k} + \underbrace{w_i^T h_{j \neq i,k} s_{j,k} + w_i^T N_k}_{N_{eff}} \tag{3.26}$$

使用(3.26)式中 Z_k、H_{eff} 及 N_{eff} 則可得等效之雜訊變異數為：

$$\sigma_{N_{eff}}^2 = w_i^H h_{j\neq i,k} h_{j\neq i,k}^H w_i + \|w_i\|^2 N_0 \tag{3.27}$$

如此(3.26)式便和在 3.3 中所介紹軟性反對映有相同的格式。注意本方法的計算複雜度跟 QAM 符元的大小和天線數成線性增加，且在這裡我們將其它根天線的殘餘干擾訊號當成雜訊處理而避免向量的解碼。

3.5 低密度同位檢查碼在 802.11n 之應用

在 802.11n[15][16]之系統中，LDPC Codes的同位檢查矩陣是由基礎同位檢查矩陣（base parity check matrix，簡稱：基礎矩陣（base matrix））H_b（附錄A）所推導而得，而不同的碼率 R_c 會對應到不同的基礎矩陣。若我們定義基礎矩陣為 $M_b \times N_b$ 的矩陣，則 N_b 表示基礎矩陣的行數，對不同碼率的基礎矩陣 N_b 皆為 24，而 M_b 表示基礎矩陣的列數，其值會隨著碼率不同而改變，可定義：

$$M_b = N_b(1-R_c) \tag{3.28}$$

，對一個大小為 $M \times N$ 同位檢查矩陣 H，我們定義：

$$z = N / N_b = M / M_b \tag{3.29}$$

，則我們可由下述之方法將 $M_b \times N_b$ 的基礎矩陣 H_b 延伸為 $M \times N$ 同位檢查矩陣 H_p：

第一步：對每一個在基礎矩陣 H_b 中不為 –1 之元素 s，我們可用一 $z \times z$ 的矩陣 I_z 來取代，其中 I_z 為向右循環位移（cyclically shift）s' 次的單位矩陣。其中位移次數 s' 為：

$$s' = s_{\mathrm{mod}(z)} \tag{3.30}$$

第二步：對每一個在基礎矩陣 H_b 中為 –1 之元素 t，我們可用一 $z \times z$ 的零矩陣(zero matrix）$0_{z\times z}$ 來取代。

很顯然的，不同的碼字長度會影響到 z 值的大小，依據 802.11n 之規格說明，碼字長度有三種可供選擇，當碼字長度為576位元時，$z=24$；碼字長度為1152位元時，$z=48$；碼字長度為1728位元時，$z=72$。一般而言，除非所傳送的封包（packet）其所含的資訊位元極少，不適用長段的區段碼，不然在使用 LDPC Codes 時，多選擇碼字長度最長者，即1728位元，這是因為由相同的基礎矩陣所推導而得的同位檢查矩陣，其中碼字長度較長者，其解碼效能會比較好，亦即位元錯誤率會較低。

3.6 低密度同位檢查碼在 802.16e 之應用

802.16e[18][19]系統之LDPC Codes同位檢查矩陣的產生方式與 802.11n系統之LDPC Codes同位檢查矩陣的產生方式相似，一樣是由一個 $M_{bm}\times N_{bm}$ 的基礎模型矩陣（base model matrix）H_{bm}（附錄B）延伸為 $M\times N$ 同位檢查矩陣 H_p。N_{bm} 亦表示為基礎模型矩陣的行數，對不同碼率的基礎模型矩陣 N_{bm} 皆為24，M_{bm} 表示基礎模型矩陣的列數，其值一樣會隨著碼率 R_c 不同而改變，亦可定義：

$$M_{bm}=N_{bm}\left(1-R_c\right) \tag{3.31}$$

，同樣地，我們可定義：

$$z_f=N/N_{bm}=M/M_{bm} \tag{3.32}$$

，我們一樣可由下述之方法將 $M_{bm}\times N_{bm}$ 的基礎模型矩陣 H_{bm} 延伸為 $M\times N$ 同位檢查矩陣 H_p。

對碼率 $R_c=2/3$ 的 A code 之 H_{bm}，其推導同位檢查矩陣 H 之過程與 802.11n 系統相同：

第一步：對每一個在基礎模型矩陣 H_{bm} 中不為 –1之元素 p_f，我們可用一 $z_f\times z_f$ 的矩陣 I_{z_f} 來取代，其中 I_{z_f} 為向右循環位移 p'_f 次的單位矩陣。其中的位移次數 p'_f

為：

$$p'_f = p_{f \bmod(z_f)} \tag{3.33}$$

第二步：對每一個在基礎模型矩陣 H_{bm} 中為 –1 之元素 q_f ，我們可用一 $z_f \times z_f$ 的零矩陣 $0_{z_f \times z_f}$ 來取代。

而對碼率 $R_c = 1/2$ 、 $R_c = 2/3$ 的 B code、 $R_c = 3/4$ 的 A、B code、 $R_c = 5/6$ 等之基礎模型矩陣 H_{bm} ，其推導同位檢查矩陣 H_p 之過程則如下述：

第一步：對每一個在基礎模型矩陣 H_{bm} 中不為 –1 之元素 p_f ，我們可用一 $z_f \times z_f$ 的矩陣 I_{z_f} 來取代，其中 I_{z_f} 為向右循環位移 p''_f 次的單位矩陣。其中位移次數 p''_f 為：

$$p''_f = \left\lfloor \frac{p_f \times z_f}{96} \right\rfloor_{\bmod(z_f)} \tag{3.34}$$

第二步：對每一個在基礎模型矩陣 H_{bm} 中為 –1 之元素 q_f ，我們可用一 $z_f \times z_f$ 的零矩陣 $0_{z_f \times z_f}$ 來取代。

與 802.11n 系統相同，不同的碼字長度會影響到 z_f 值的大小，依據 802.16e 之規格說明，802.16e 系統在經過連鎖（concatenation）後會依據傳送資訊所使用的調變、編碼器、及其碼率來決定傳送區塊（block）的大小，而碼字長度再依照區塊的大小來決定，如表 3-1 所示，其中的參數“k”之值即為區塊的大小，單位為 byte，當區塊大小被決定後，即可由表 3-1 來確定碼字長度與參數“ z_f ”之值的大小，與 802.11n 系統一樣，原則上，儘可能的選用碼字長度最長者來產生同位檢查矩陣，來進行編碼，理論上由相同的基礎模型矩陣所推導而得的同位檢查矩陣，其中碼字長度較長者，其解碼效能較好，位元錯誤率較低。

n (bit)	n (byte)	z_f	k (byte)				Number of subchannels		
			R=1/2	R=2/3	R=3/4	R=5/6	QPSK	16QAM	64QAM
579	72	24	36	48	54	60	6	3	2
672	84	28	42	56	63	70	7	-	-
768	96	32	48	64	72	80	8	4	-
864	108	36	54	72	81	90	9	-	3
960	120	40	60	80	90	100	10	5	-
1056	132	44	66	88	99	110	11	-	-
1152	144	48	72	96	108	120	12	6	4
1248	156	52	78	104	117	130	13	-	-
1344	168	56	84	112	126	140	14	7	-
1440	180	60	90	120	135	150	15	-	5
1536	192	64	96	128	144	160	16	8	-
1632	204	68	102	136	153	170	17	-	-
1728	216	72	108	144	162	180	18	9	6
1824	228	76	114	152	171	190	19	-	-
1920	240	80	120	160	180	200	20	10	-
2016	252	84	126	168	189	210	21	-	7
2112	264	88	132	176	198	220	22	11	-
2208	276	92	138	184	207	230	23	-	-
2304	288	96	144	192	216	240	24	12	8

表 3-1 LDPC Codes 之區塊大小和碼率表

3.7 通道模型

3.7.1 802.11n 通道模型

TGn Sync 提供了 A, B, C, D, E, F,六種通道模型供 802.11n 系統模擬使用，而我們只使用通道 B（NLOS：non-line-of-sight）：距離 6m，通道 D（NLOS）：距離 11m，通道 D（NLOS）：距離 21m，來作為系統的通道模擬環境，較詳細的通道特性如表 3-2 和表 3-3 所示。其中，在 line-of-sight（LOS）的情形下，K-factor

只會對應到第一根 tap，而 K-factor = -∞則對應到其它的 tap。另外，參數 d_{BP} 則稱為臨界距離（breaking point distance），若通道模型中，傳送端與接收端之間的距離小於 d_{BP}，則為 LOS 通道；若傳送端與接收端之間的距離大於 d_{BP}，則為 NLOS 通道。

Model	Condition	K-factor (dB)	RMS delay spread (ns)	# of clusters
B	LOS/NLOS	0 / -∞	15	2
D	LOS/NLOS	3 / -∞	50	3
E	LOS/NLOS	6 / -∞	100	4

表 3-2 通道模型 LOS/NLOS 參數

Model	d_{BP} (m)	Slope before d_{BP}	Slope after d_{BP}	Shadow fading std. dev. (dB) before d_{BP} (LOS)	Shadow fading std. dev. (dB) after d_{BP} (NLOS)
B	5	2	3.5	3	4
D	10	2	3.5	3	5
E	20	2	3.5	3	6

表 3-3 通道模型路徑損失（path loss）參數

3.7.2 802.16e 通道模型

IEEE 802.16e 系統的規格說明上並未提供模擬使用之通道模型，在此我們使用了 3GPP 所提供的通道模型，以作為模擬使用。由於 3G 系統的通道模型環境，

是定義在無線都會型區域（wireless metropolitan area）中，且可支援定點（fixed）與行動（mobile）無線多重路徑（wireless multipath fading）通道，故此一通道模型環境與 IEEE 802.16e 系統之操作的環境十分相似，可使用於 IEEE 802.16e 系統的模擬上。其中，3GPP 之通道模型有三種形式：

1. Suburban macro-cell：cell 涵蓋範圍為 1～6Km，BS（base station）端天線置於高於建築物處，範圍為 10m 至 80m，平均高度為 32m。SS（subscriber station）端移動速度介於 0～250 Km/hr。
2. Urban macro-cell：cell 涵蓋範圍為 1～6Km，BS 端天線置於高於建築物處，範圍為 10m 至 80m，平均高度為 32m。SS 端移動速度介於 0～250 Km/hr。Urban macro-cell 在上述之環境與 Suburban macro-cell 之環境相近，但在其它部份可能與 Suburban macro-cell 之環境略有不同。
3. Urban micro-cell：BS 與 SS 之間的距離約為 1Km，cell 涵蓋範圍為 0.3～0.5Km，BS 端天線置於建築物之上，平均高度為 12.5m。SS 端移動速度介於 0～120 Km/hr。

而我們是選用 Urban macro-cell 的通道模型，作為我們模擬測試的通道環境。

3.8 模擬結果

3.8.1 802.11n 系統模擬結果

3.8.1 的模擬結果是使用本章所提到的 MMSE 偵測及軟性反對映來對接收訊號進行偵測與軟性輸出的計算，並在依據 802.11n 規格說明所建立的 2×2、4×4 通道 B、D、E（NLOS）上測試所得之結果。其中的 LDPC Codes 解碼器，則是使用第 2 章所介紹的演算法，包括 Sum-product 演算法（SPA）、Normalized BP-based 演算法（NBP）、Normalized APP-based 演算法（NAPP）和 Layered Normalized BP-based 演算法（LNBP）四種不同的演算法。而模擬中所選用的

Modulation-Coding Scheme（MCS）則有 2 根傳送天線的 MCS11、MCS13、MCS15，和 4 根傳送天線的 MCS27、MCS28，詳細的 MCS11、MCS13、MCS15、MCS27、MCS28 規格如表 3-4 所示。

MCS index	Number of spatial streams	Modulation	Coding rate	GI = 800ns		GI = 400ns	
				Rate in 20MHz	Rate in 40MHz	Rate in 20MHz	Rate in 40MHz
11	2	16-QAM	1/2	52.00	108.00	57.78	120.00
13	2	64-QAM	2/3	104.00	216.00	115.56	240.00
15	2	64-QAM	5/6	130.00	270.00	144.44	300.00
27	4	16-QAM	1/2	104.00	216.00	115.56	240.00
28	4	16-QAM	3/4	156.00	324.00	173.33	360.00

表 3-4 802.11n 之 Modulation-Coding Scheme 表

圖 3-7 到圖 3-12 都是使用 Sum-product 演算法，在估計通道（estimation channel）並假設完美同步（perfect synchronization）下的模擬結果，其中我們設定最大遞迴次數為 50，也就是說當遞迴次數為 50 時，即使 $H_p\vec{V}^T \neq 0$ 仍強制將解碼所得之結果送出。

圖 3-13 到圖 3-30 分別是使用 Normalized BP-based 演算法（圖 3-13 到圖 3-18）、Normalized APP-based 演算法（圖 3-19 到圖 3-24）和 Layered Normalized BP-based 演算法（圖 3-25 到圖 3-30），在估計通道（estimation channel）並包含部份非理想效應，最大遞迴次數為 15 次下的模擬結果。其中非理想效應有載波頻率偏移（carrier frequency offset，CFO）、取樣頻率偏移（sample frequency offset，SFO）、直流偏移（DC offset）、I-Q 不平衡（I-Q imbalance）、相位雜訊（phase noise）、非線性放大失真（nonlinear amplifier distortion）。

由圖 3-13 到圖 3-30 中，我們可看出使用 Layered Normalized BP-based 演算法解碼的效能較好，位元錯誤率較低，其次是 Normalized BP-based 演算法，最差的則是 Normalized APP-based 演算法。但從運算複雜度的觀點上來看，Normalized APP-based 演算法的運算復雜度卻是最低的，而 Normalized BP-based 演算法和 Layered Normalized BP-based 演算法兩者的運算復雜度則大致上相同。雖然 Layered Normalized BP-based 演算法因為其分層解碼的架構破壞了其平行處理結構，但其收斂速度卻幾乎是 Normalized BP-based 演算法的兩倍，舉例來說：如果 Normalized BP-based 演算法最大遞迴次數為 N_{ite}，那麼 Layered Normalized BP-based 演算法的最大遞迴次數大約只要 $\frac{N_{ite}}{2}$ 次即可達到相近的位元錯誤率。另外，也因為此分層解碼的架構使得 Layered Normalized BP-based 演算法在和 Normalized APP-based 演算法一樣，在記憶體的需求上只需 Normalized BP-based 演算法的一半，在晶片設計的繞線（routing）上也更為容易。

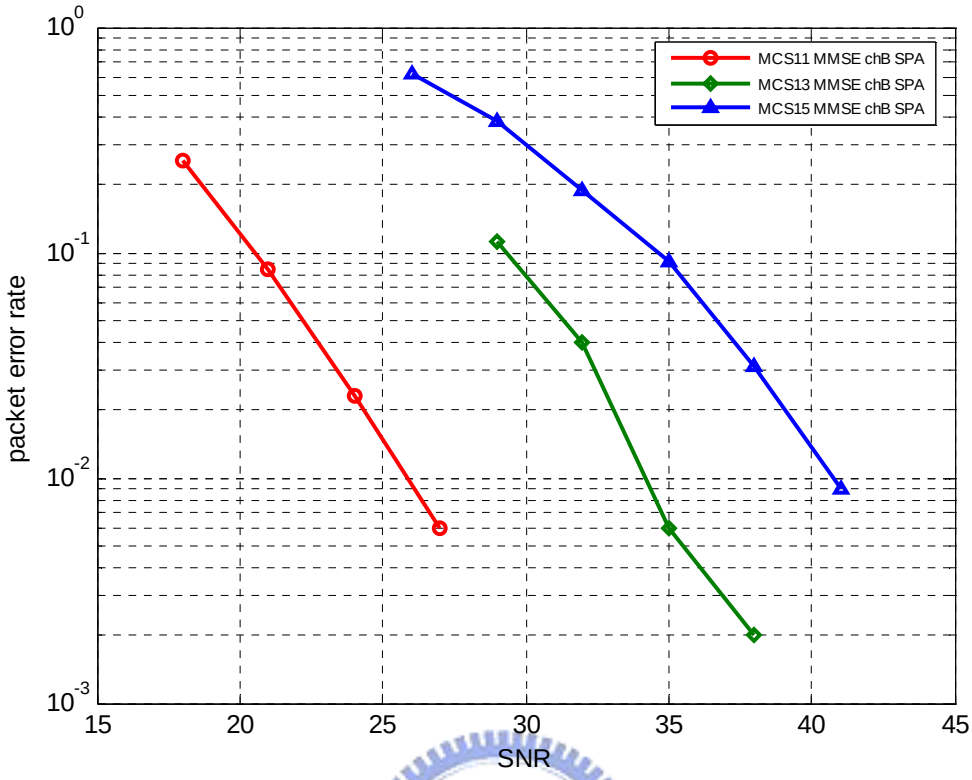


圖 3-7 SPA 解碼在 2×2 通道 B 下模擬結果

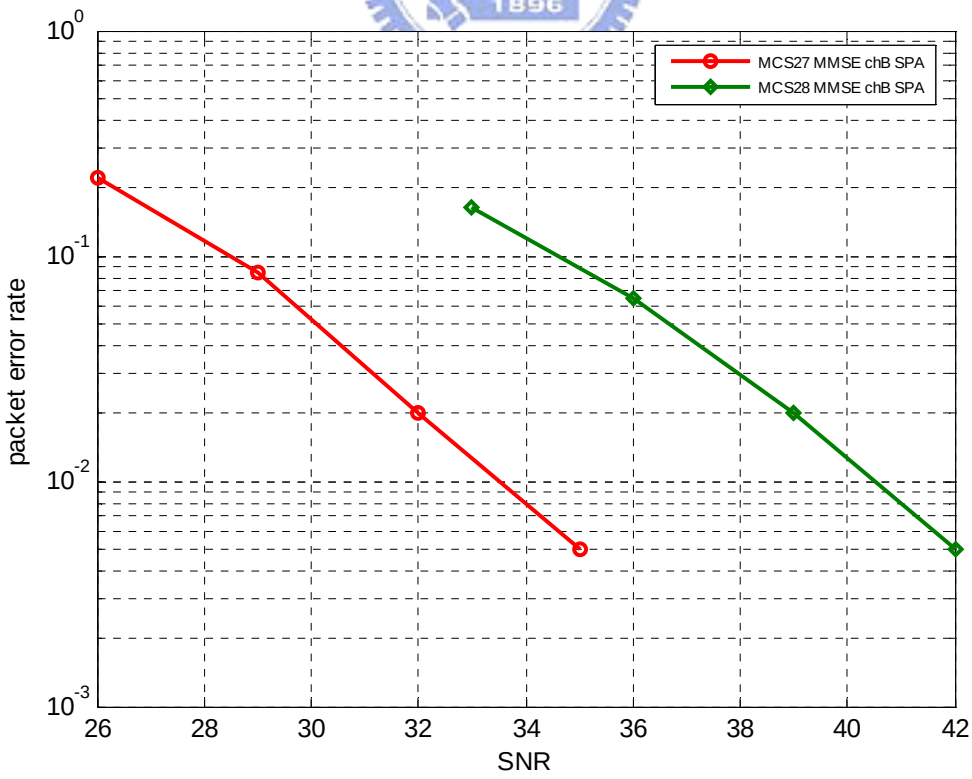


圖 3-8 SPA 解碼在 4×4 通道 B 下模擬結果

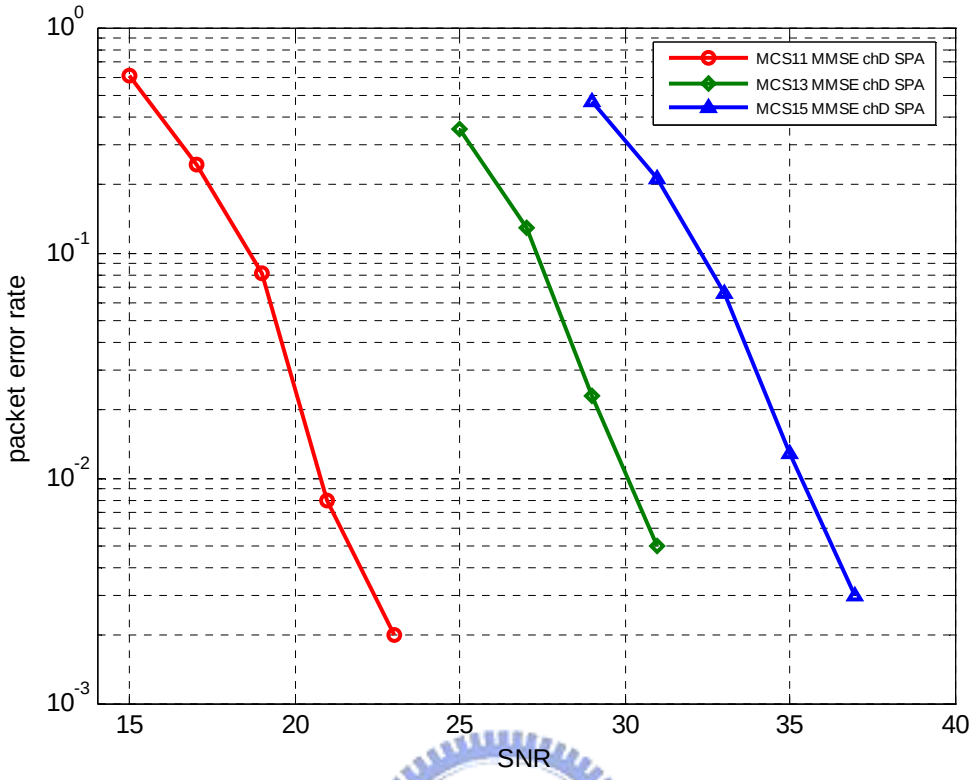


圖 3-9 SPA 解碼在 2×2 通道 D 下模擬結果

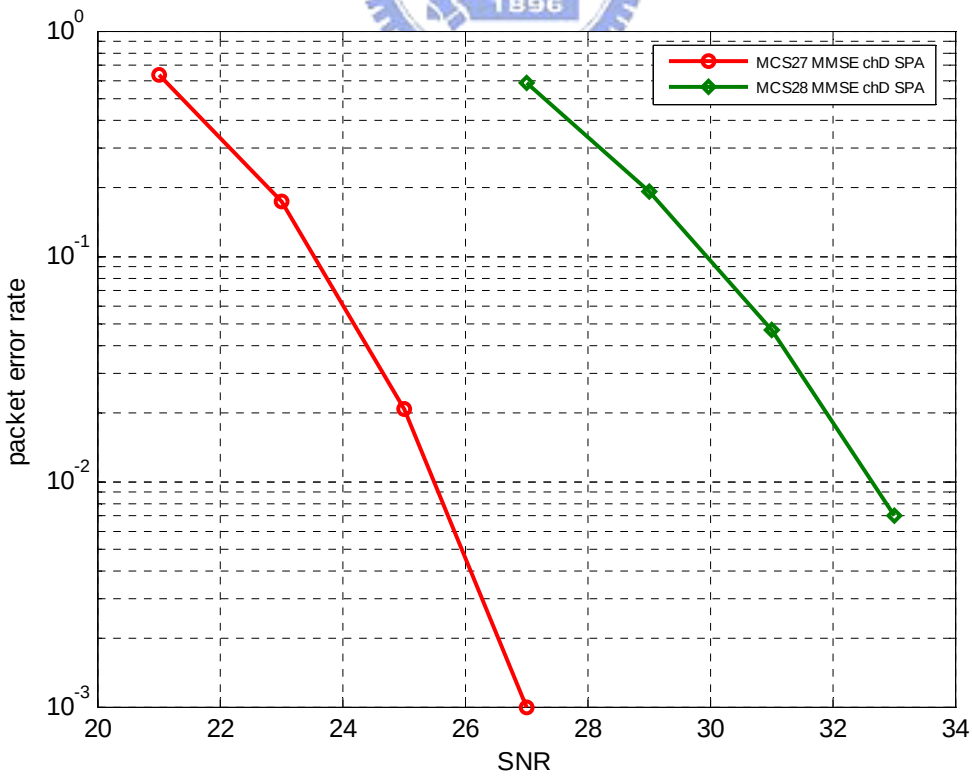


圖 3-10 SPA 解碼在 4x4 通道 D 下模擬結果

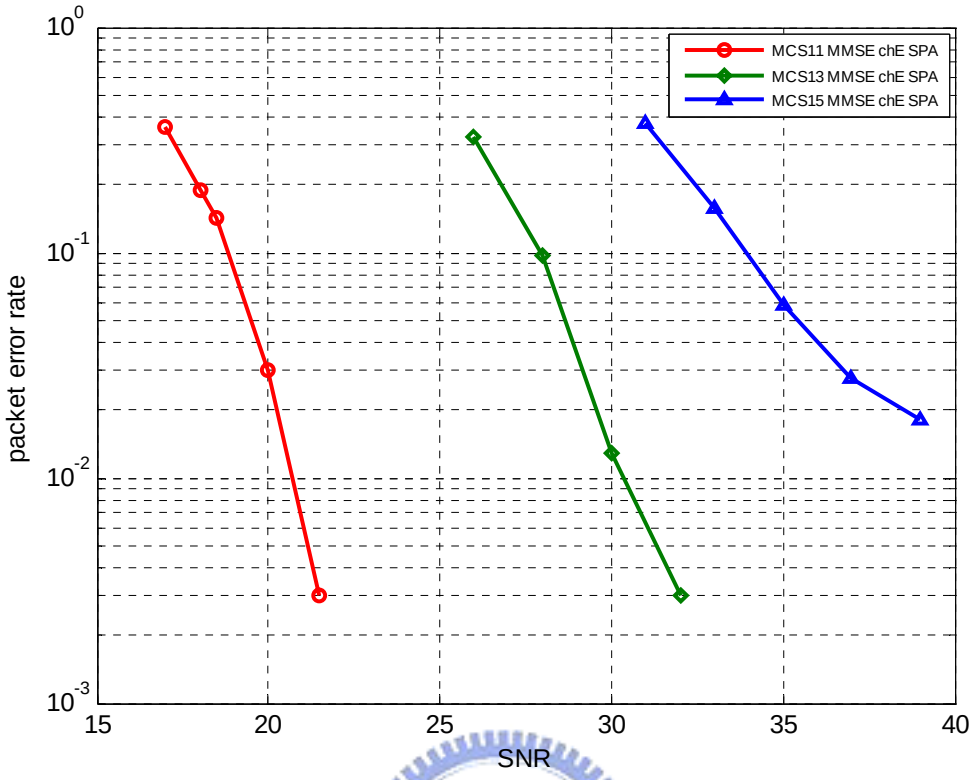


圖 3-11 SPA 解碼在 2x2 通道 E 下模擬結果

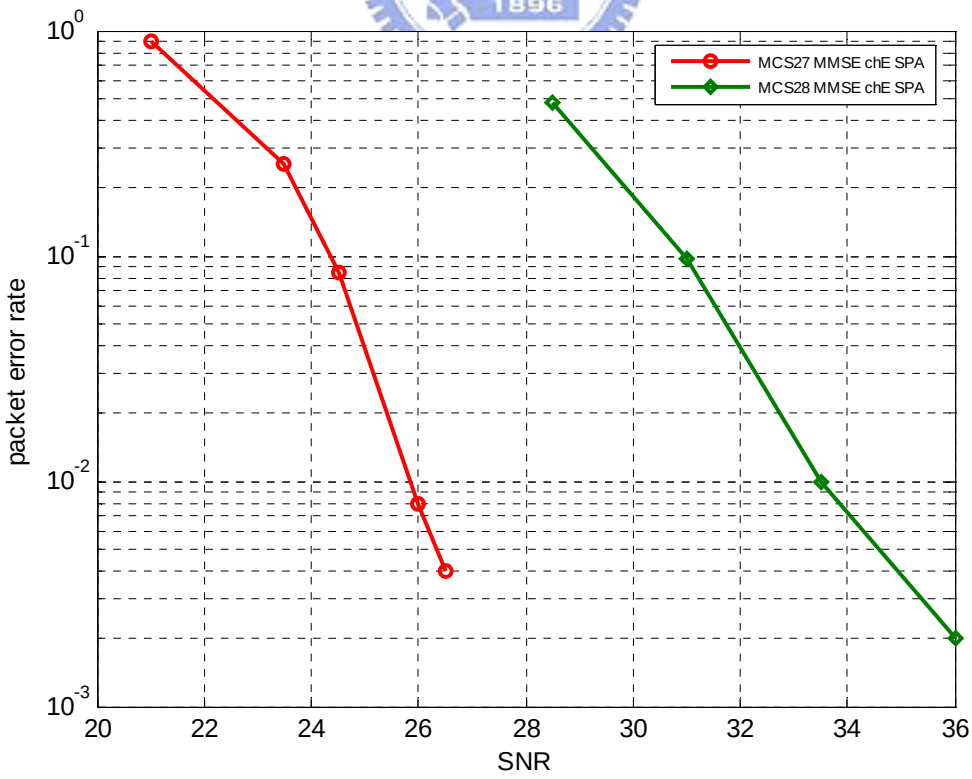


圖 3-12 SPA 解碼在 4x4 通道 E 下模擬結果

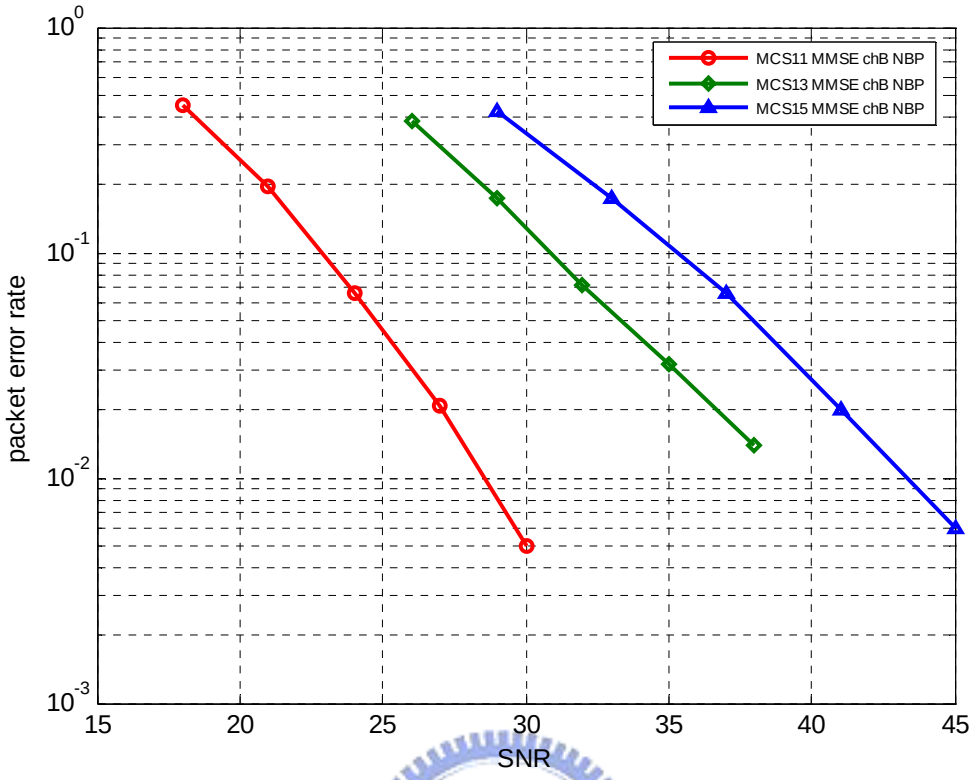


圖 3-13 NBP 解碼在 2x2 通道 B 下模擬結果

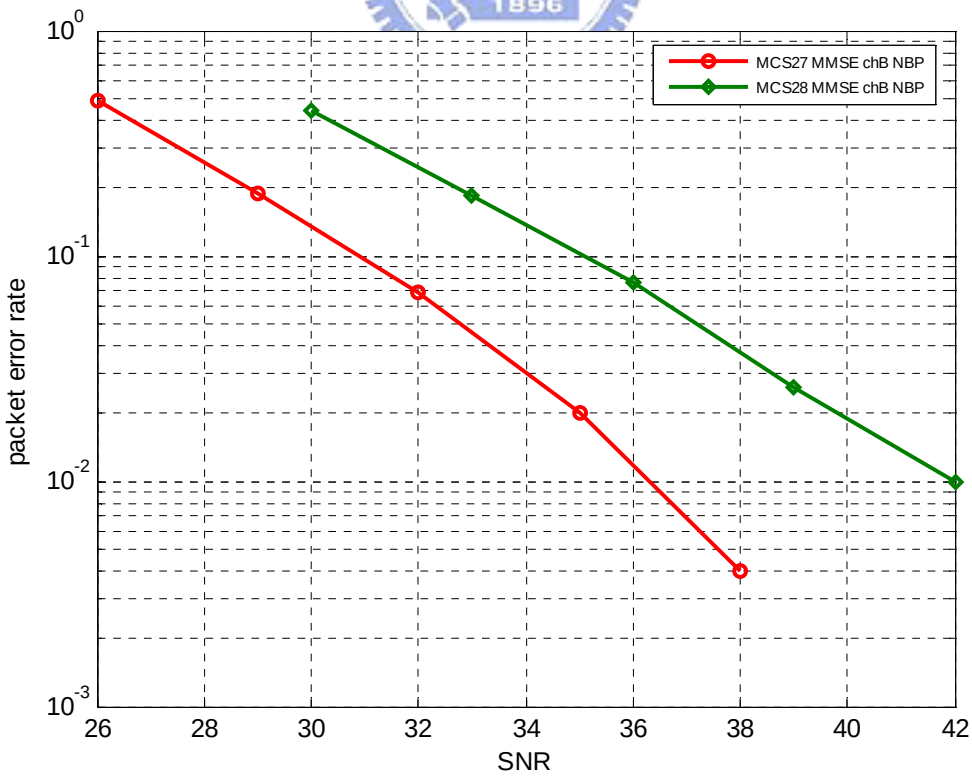


圖 3-14 NBP 解碼在 4x4 通道 B 下模擬結果

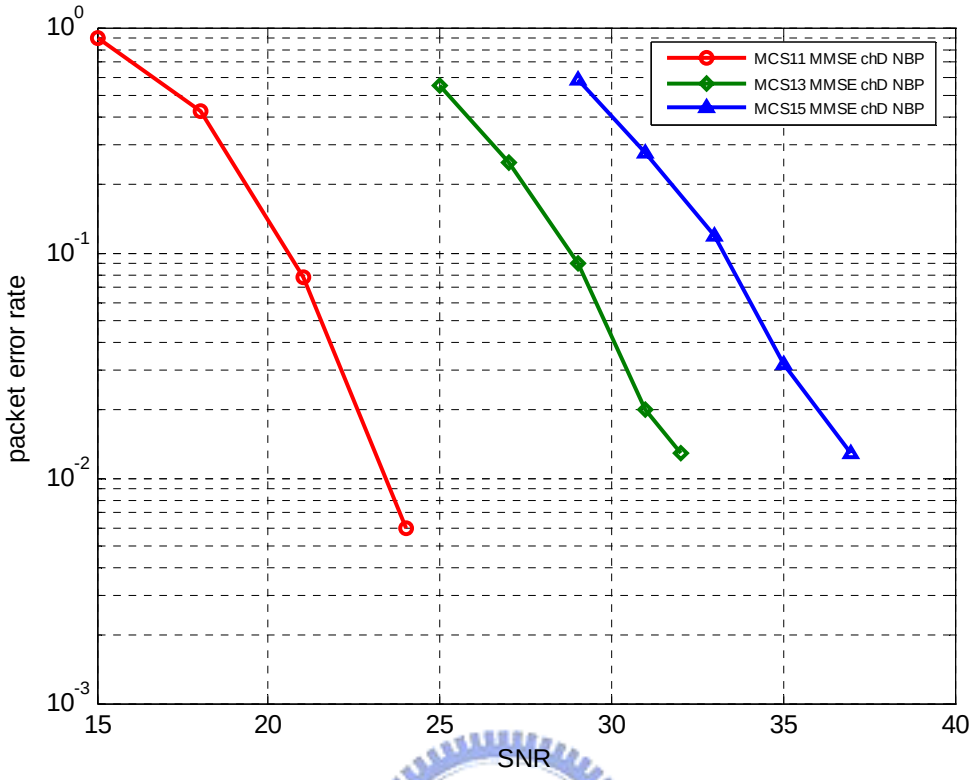


圖 3-15 NBP 解碼在 2x2 通道 D 下模擬結果

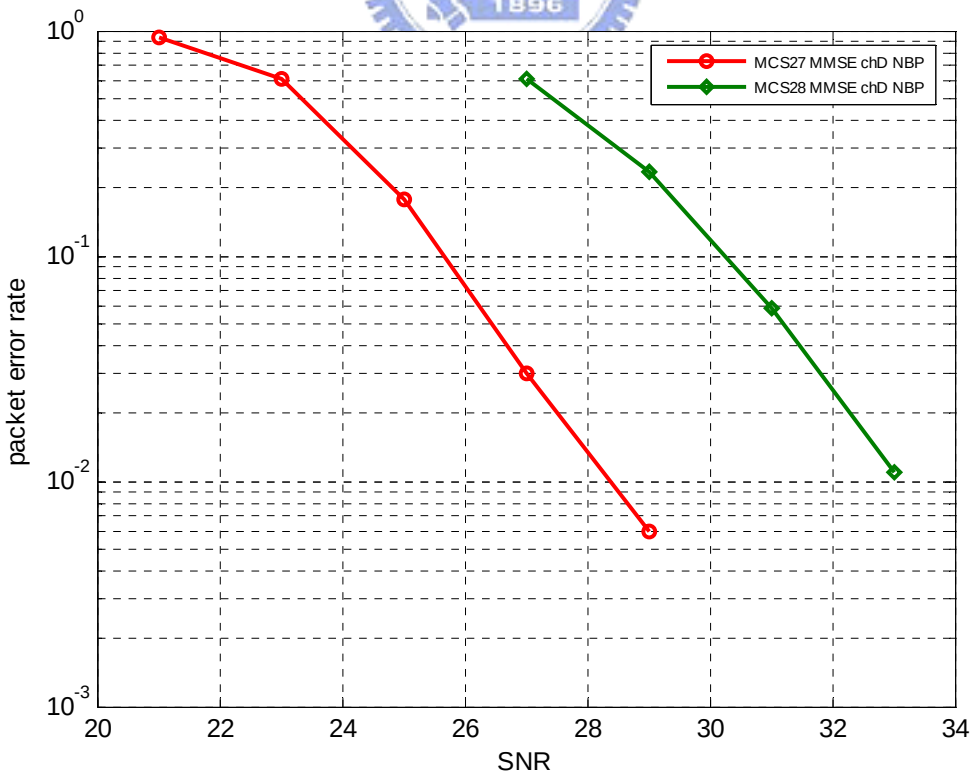


圖 3-16 NBP 解碼在 4x4 通道 D 下模擬結果

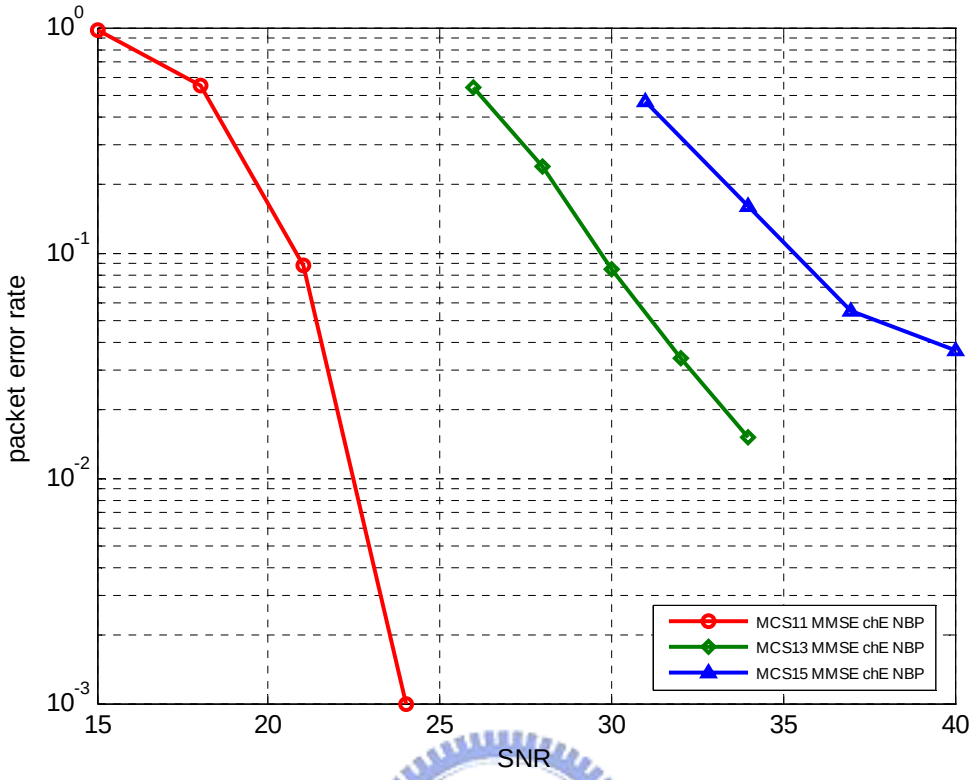


圖 3-17 NBP 解碼在 2x2 通道 E 下模擬結果

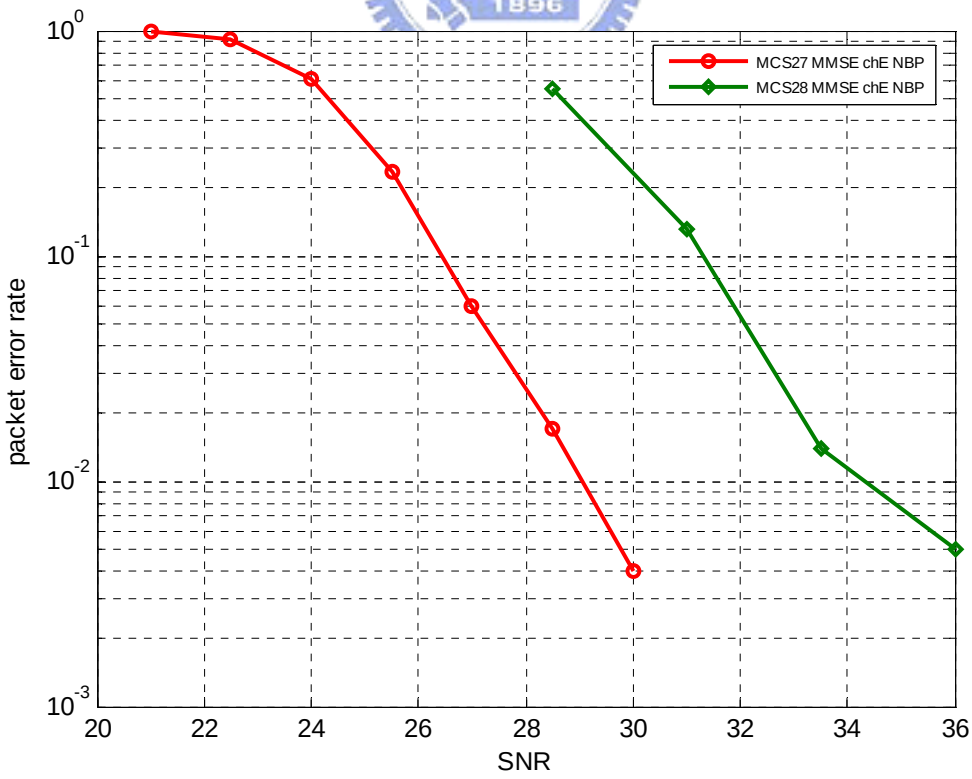


圖 3-18 NBP 解碼在 4x4 通道 E 下模擬結果

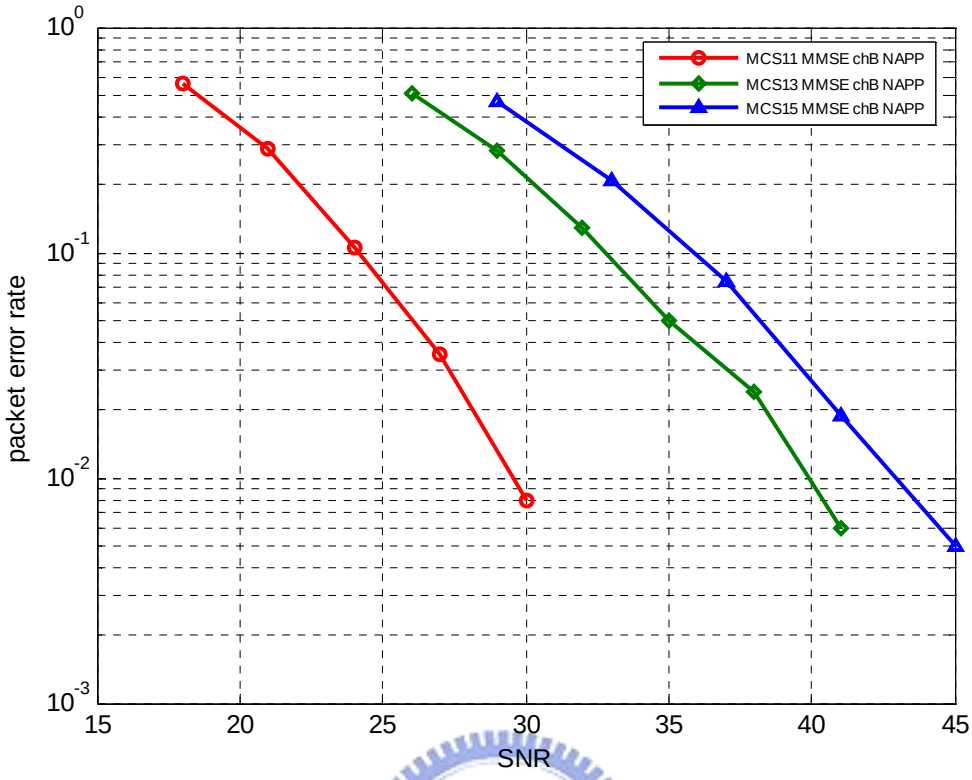


圖 3-19 NAPP 解碼在 2x2 通道 B 下模擬結果

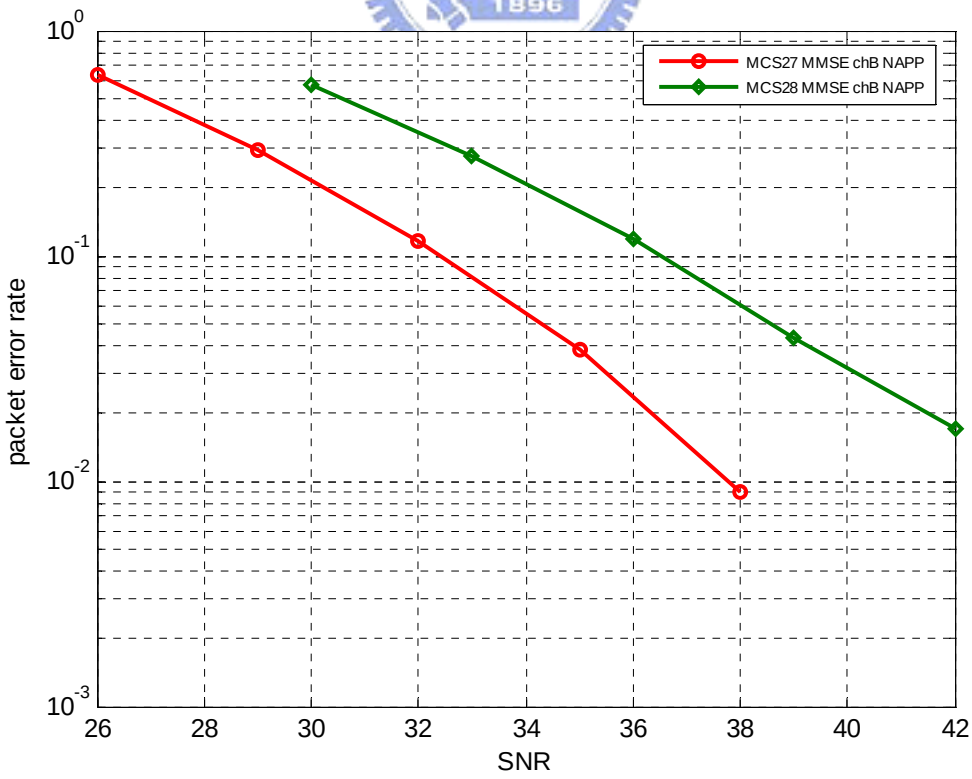


圖 3-20 NAPP 解碼在 4x4 通道 B 下模擬結果

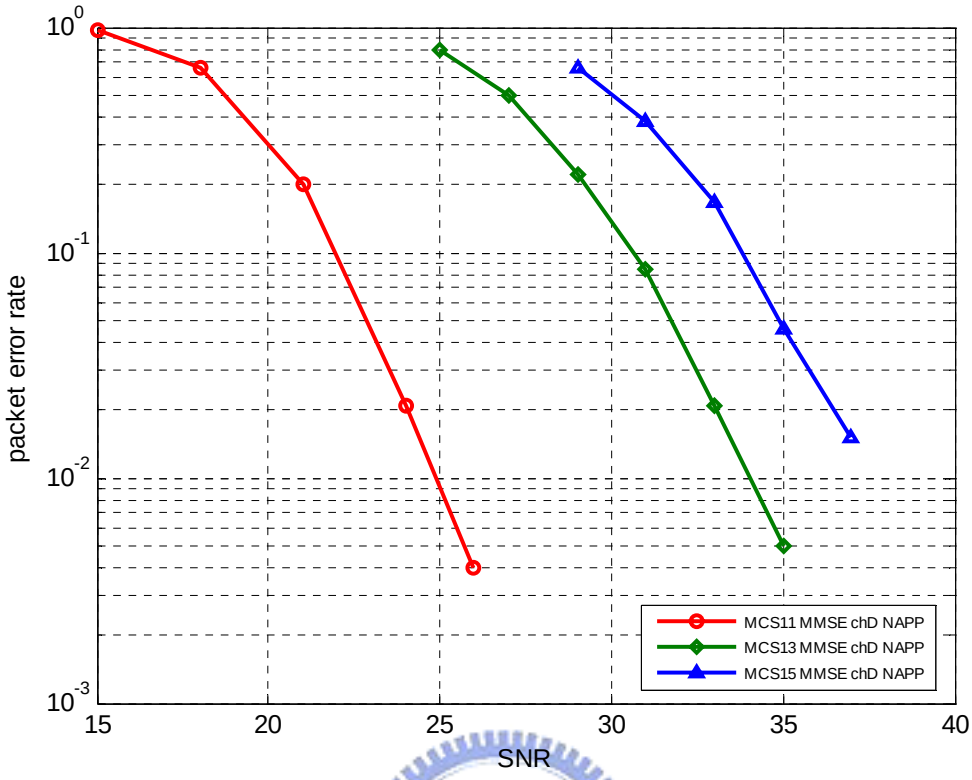


圖 3-21 NAPP 解碼在 2x2 通道 D 下模擬結果

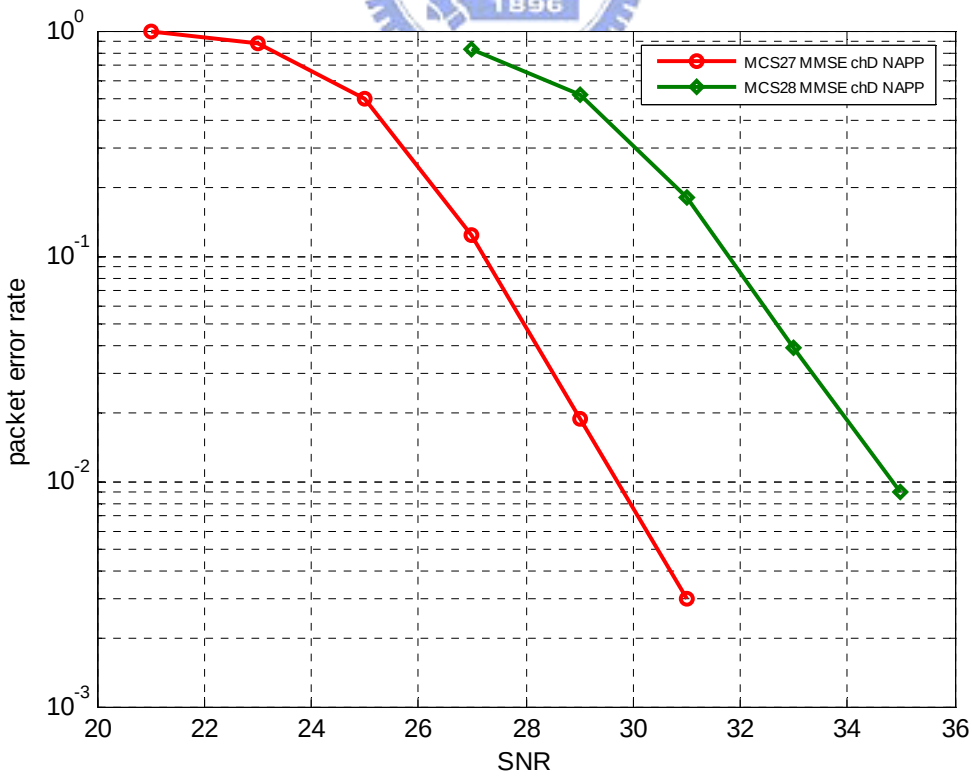


圖 3-22 NAPP 解碼在 4x4 通道 D 下模擬結果

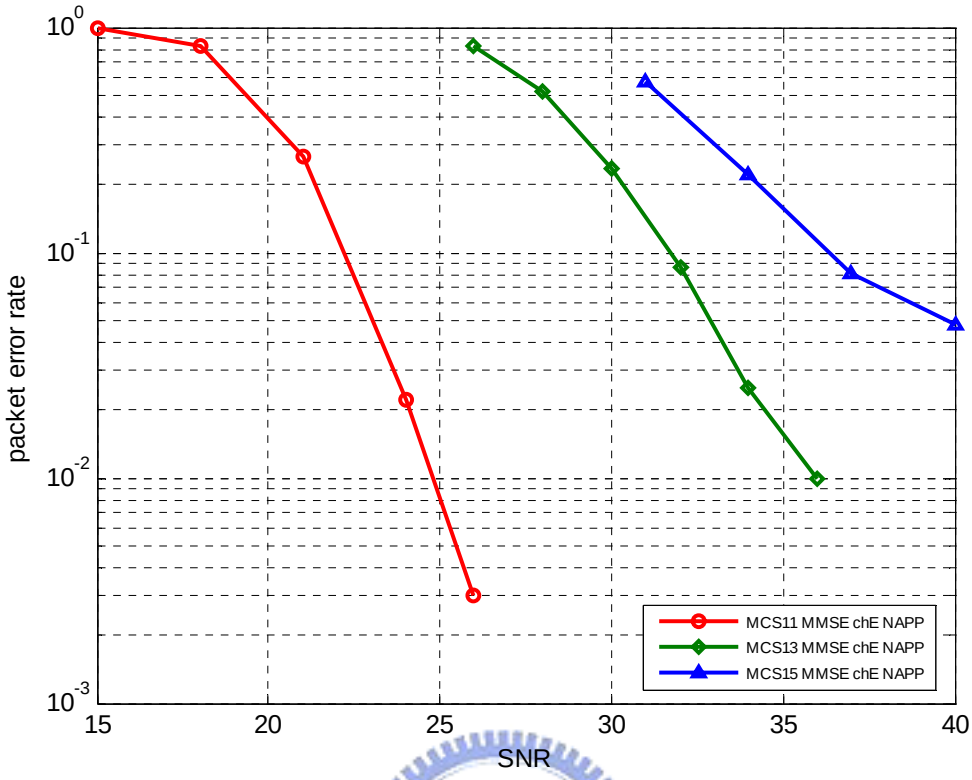


圖 3-23 NAPP 解碼在 2x2 通道 E 下模擬結果

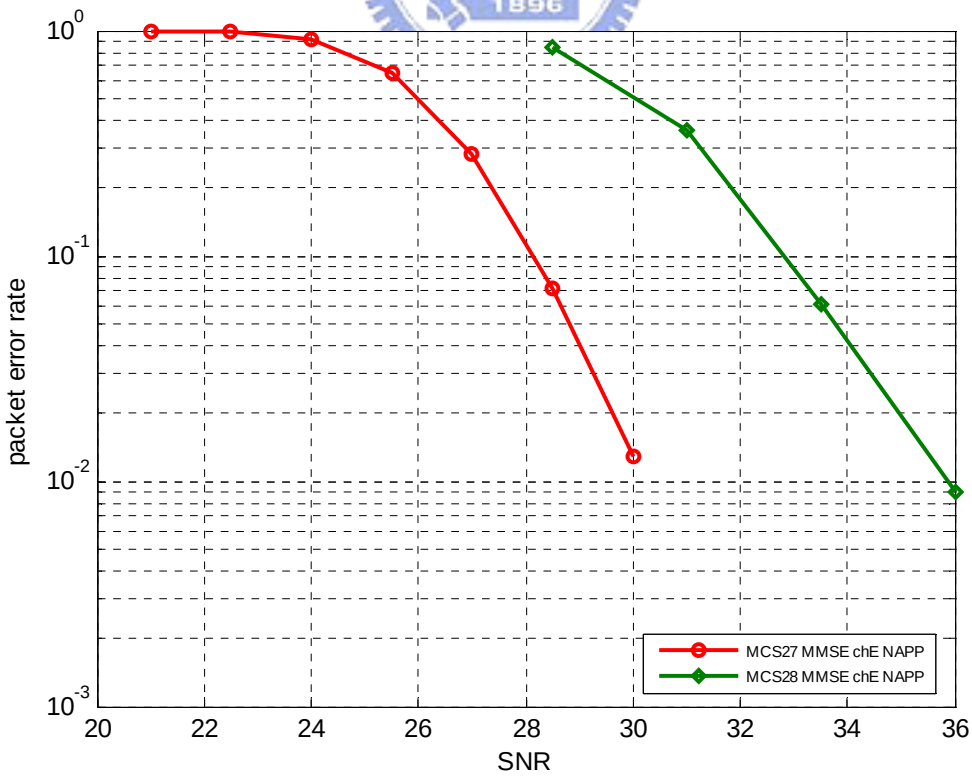


圖 3-24 NAPP 解碼在 4x4 通道 E 下模擬結果

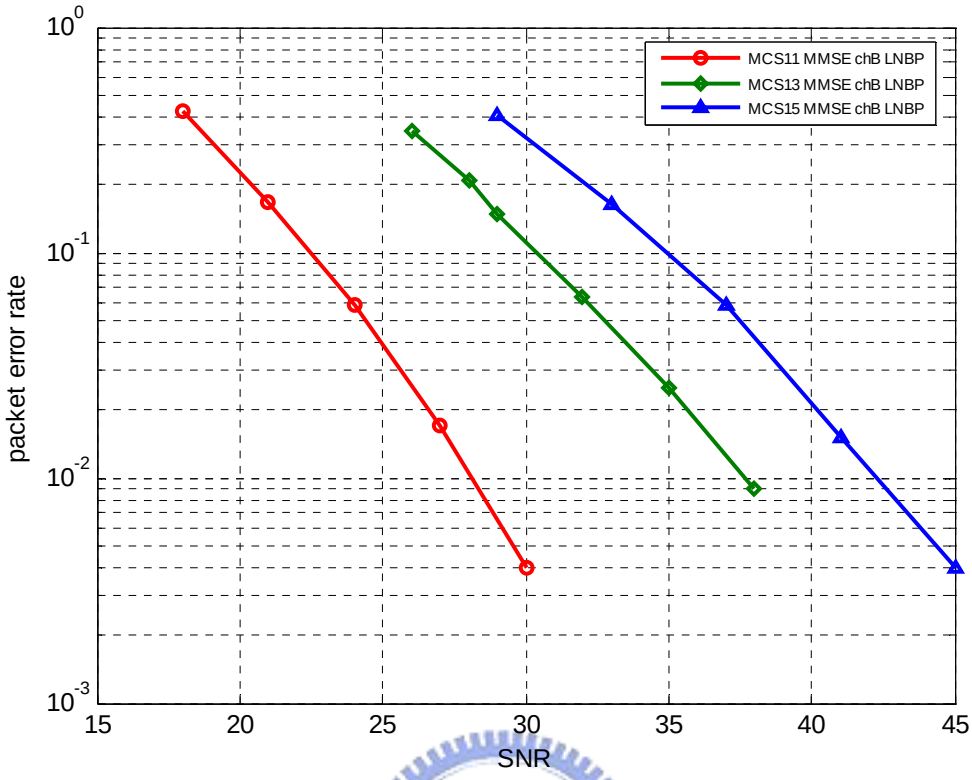


圖 3-25 LNBP 解碼在 2x2 通道 B 下模擬結果

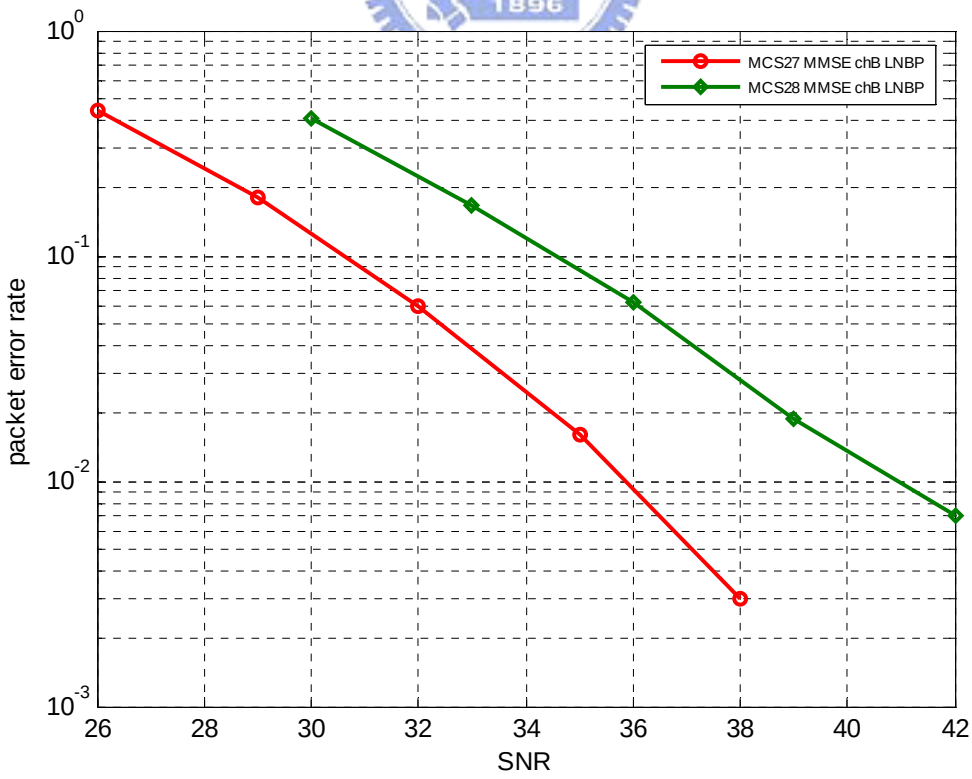


圖 3-26 LNBP 解碼在 4x4 通道 B 下模擬結果

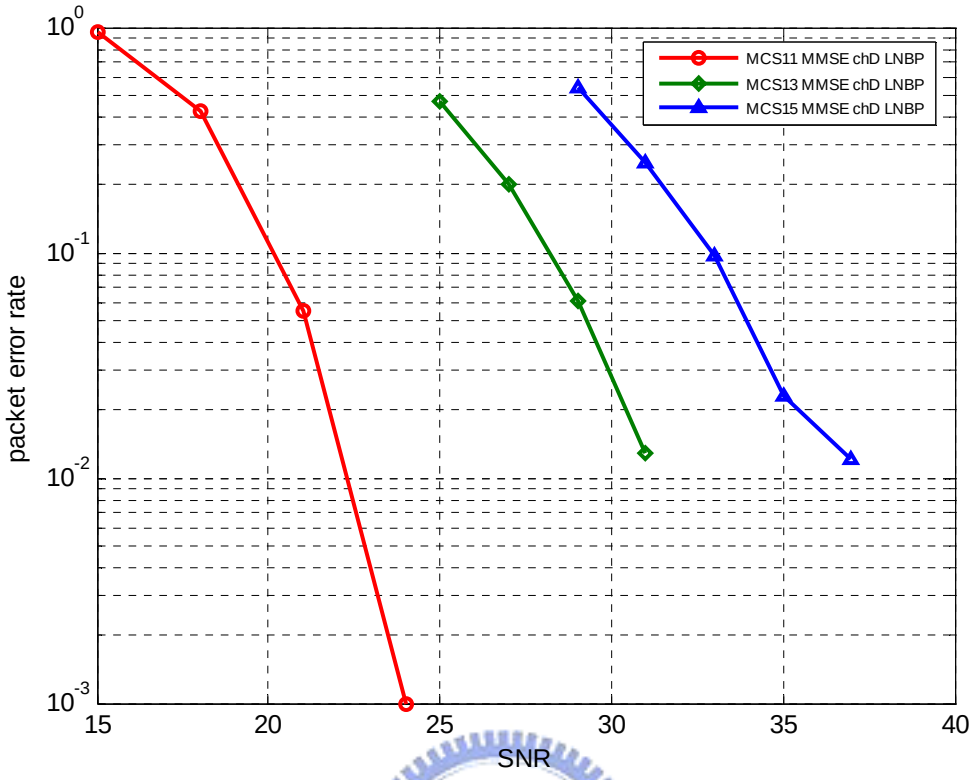


圖 3-27 LNBP 解碼在 2x2 通道 D 下模擬結果

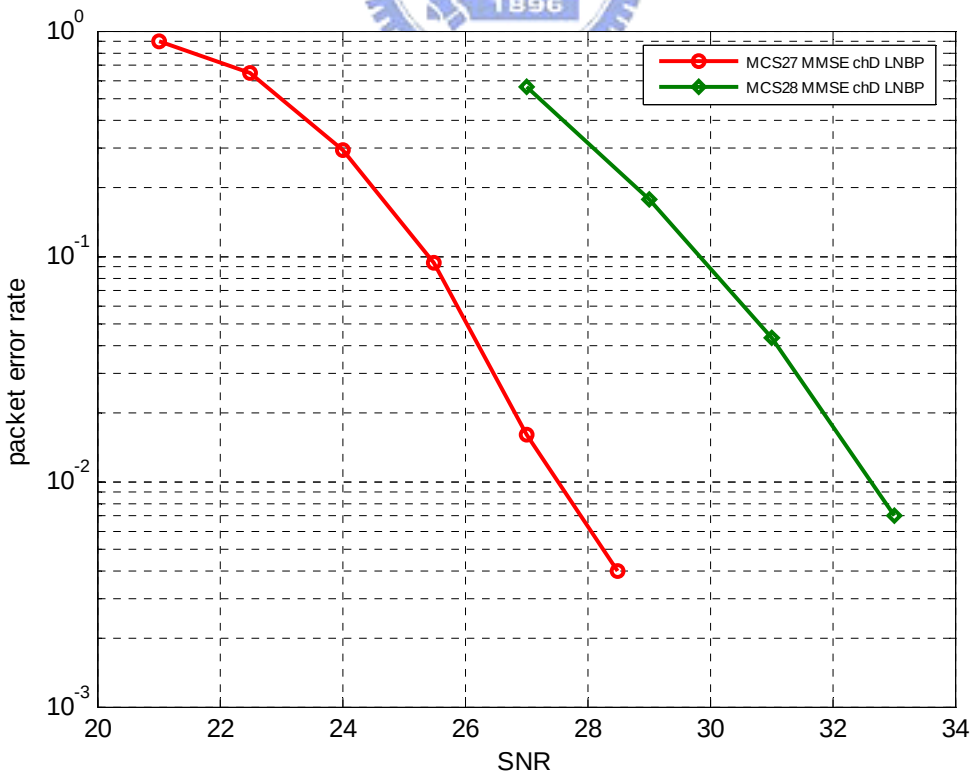


圖 3-28 LNBP 解碼在 4x4 通道 D 下模擬結果

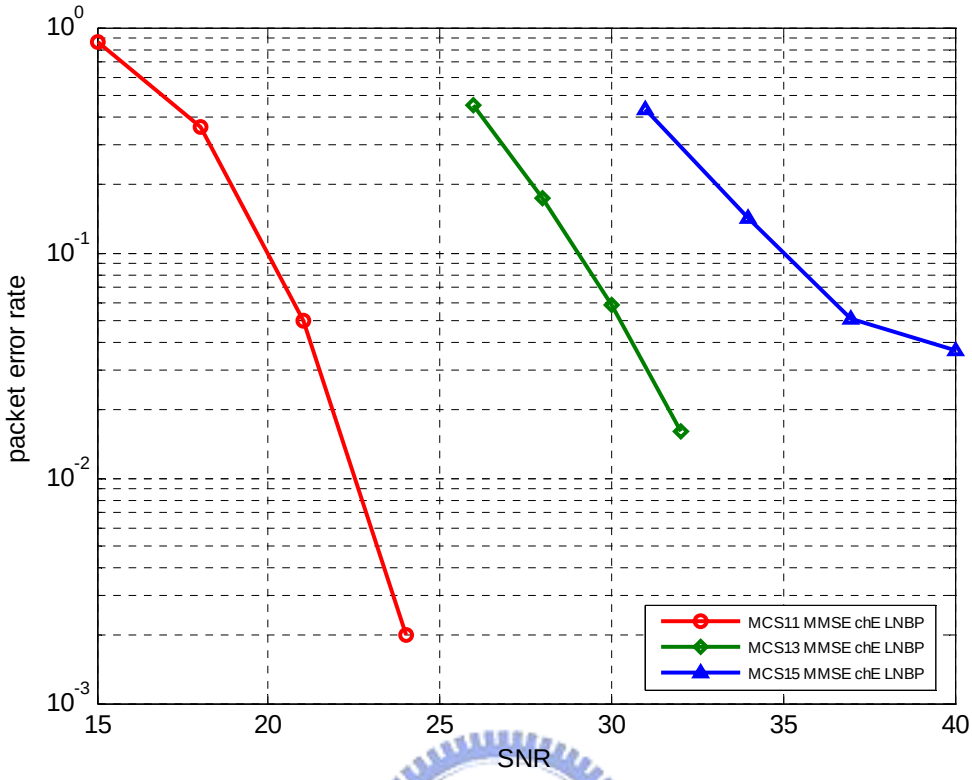


圖 3-29 LNBP 解碼在 2x2 通道 E 下模擬結果

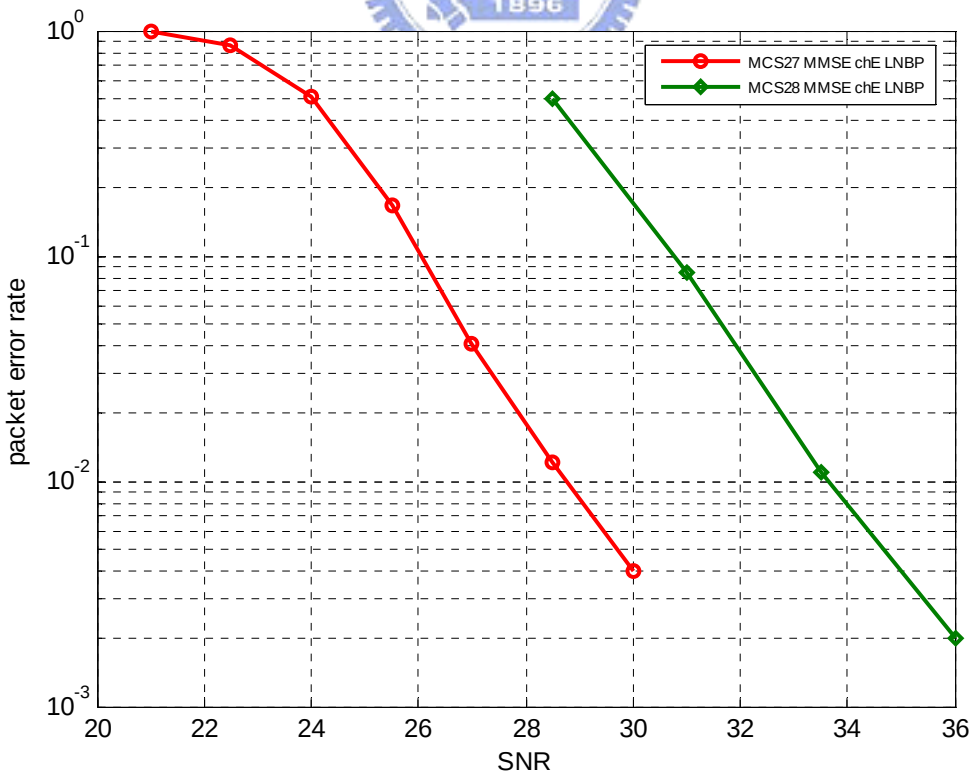


圖 3-30 LNBP 解碼在 4x4 通道 E 下模擬結果

3.8.2 802.16e 系統模擬結果

3.8.2 的模擬結果是使用本章所提到的 MMSE 偵測及軟性反對映來對接收訊號進行偵測與軟性輸出的計算，並在 3GPP 所提供的通道上測試所得之結果。其中的編碼器是使用碼率1/2 、碼字長度1152和576位元的 LDPC Codes，而解碼器，亦是使用第 2 章所介紹的演算法，包括 Normalized BP-based 演算法（NBP）、Normalized APP-based 演算法（NAPP）和 Layered Normalized BP-based 演算法（LNBP）三種不同的演算法，並設定最大遞迴次數為15次。

圖 3-31 到圖 3-33 為 QPSK 調變下的模擬結果，圖 3-34 到圖 3-36 為 16QAM 調變下的模擬結果，而圖 3-37 到圖 3-39 則為 64QAM 調變下的模擬結果。其中，圖 3-31、圖 3-34、和圖 3-37 為理想通道（perfect channel）下的模擬結果；圖 3-32、圖 3-35、和圖 3-38 為使用估計通道（estimation channel）下的模擬結果，我們以 ec 來表示；而圖 3-33、圖 3-36、和圖 3-39 則為使用估計通道並加上載波頻率偏移（carrier frequency offset，CFO）下的模擬結果，我們以 ec cfo 來表示。最後，為了方便觀察，我們將使用相同 LDPC Codes 解碼演算法的模擬結果放置於同一張圖上比較，圖 3-40 為使用 Normalized BP-based 演算法的模擬結果，圖 3-41 為使用 Normalized APP-based 演算法的模擬結果，圖 3-42 則為使用 Layered Normalized BP-based 演算法的模擬結果。

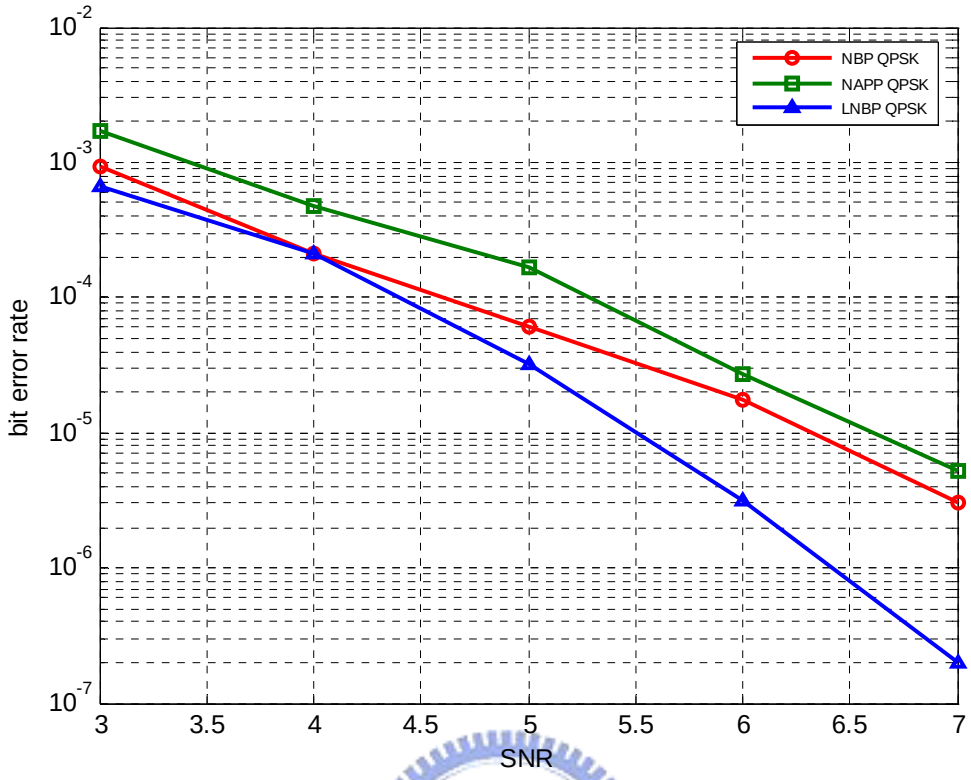


圖 3-31 802.16e 系統在 QPSK 調變、理想通道下模擬結果

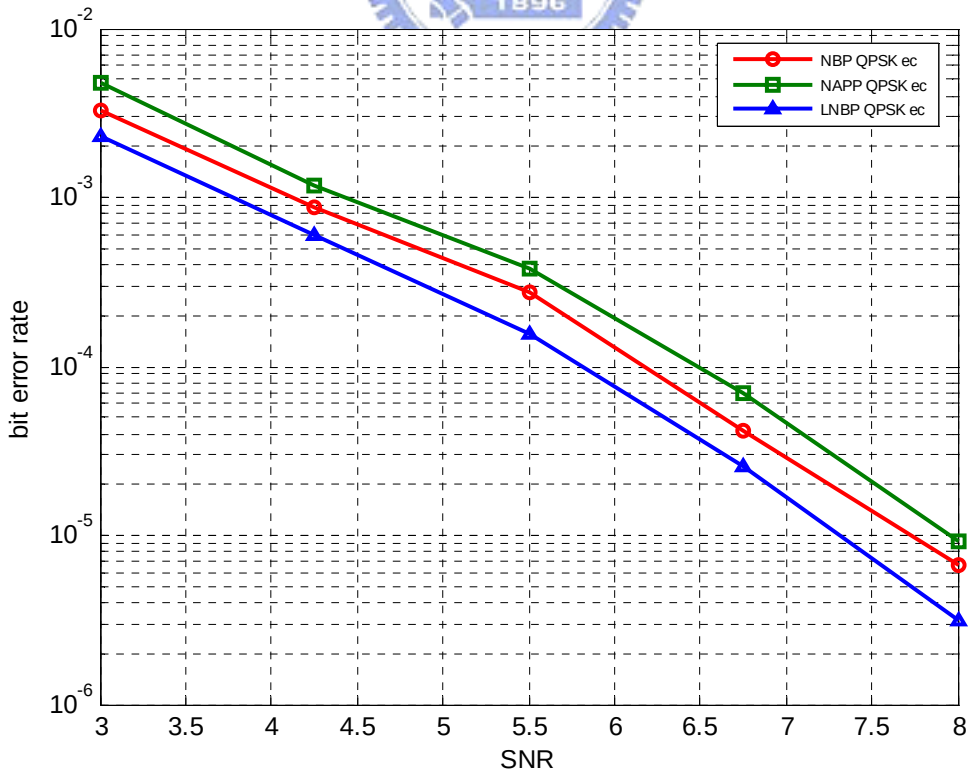


圖 3-32 802.16e 系統在 QPSK 調變、估計通道下模擬結果

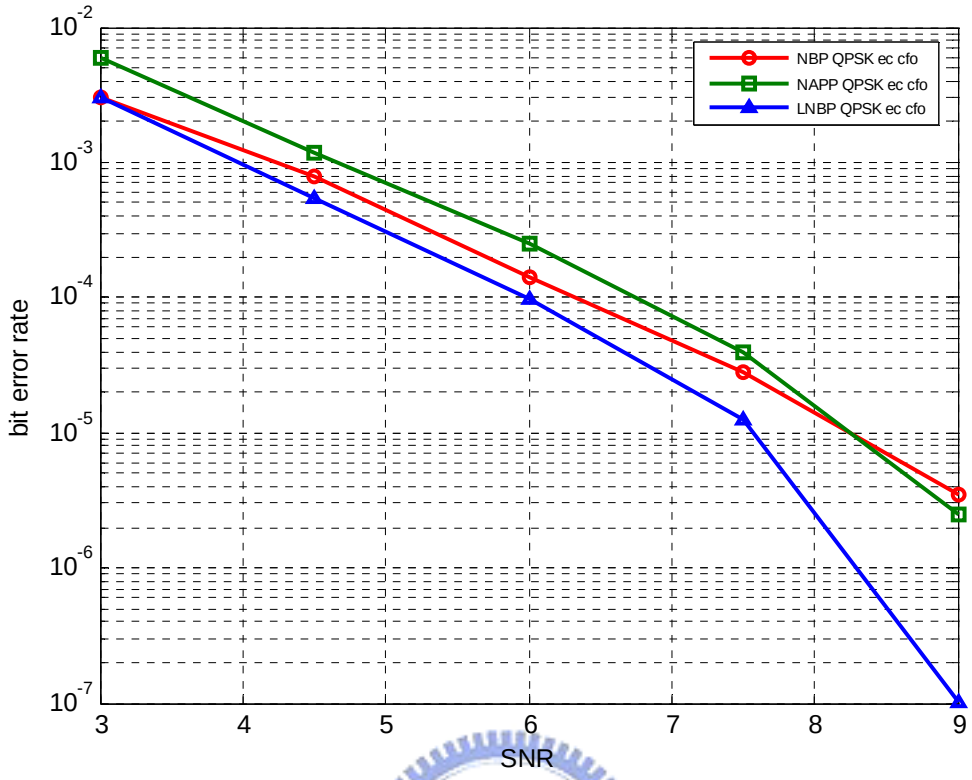


圖 3-33 802.16e 系統在 QPSK 調變、估計通道、加入 CFO 效應下模擬結果

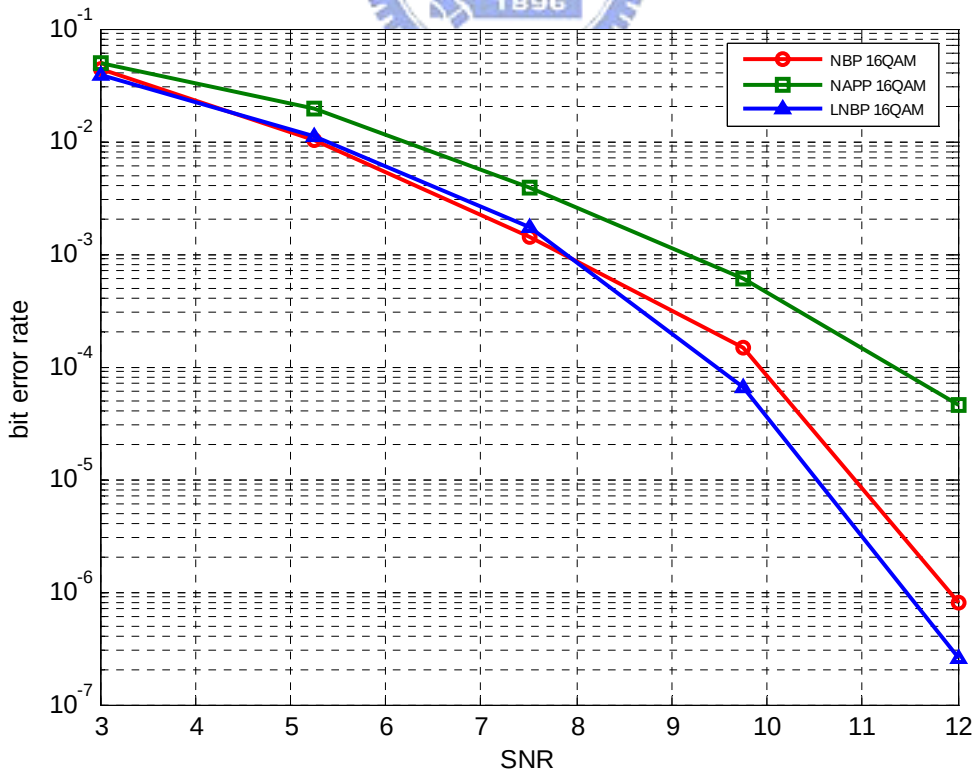


圖 3-34 802.16e 系統在 16QAM 調變、理想通道下模擬結果

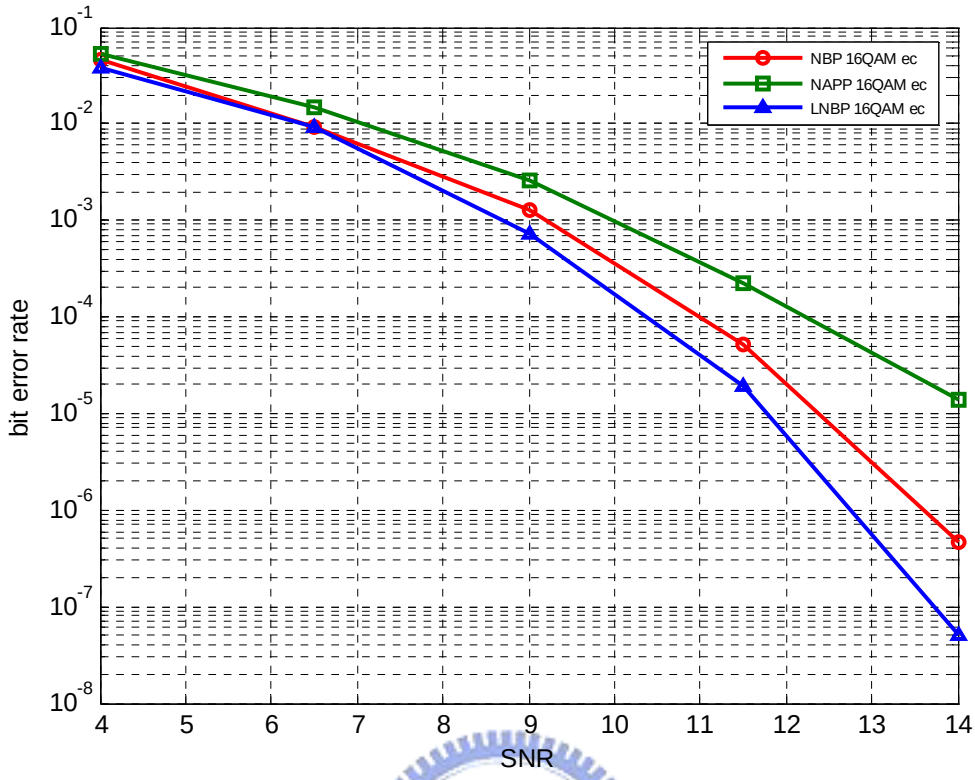


圖 3-35 802.16e 系統在 16QAM 調變、估計通道下模擬結果

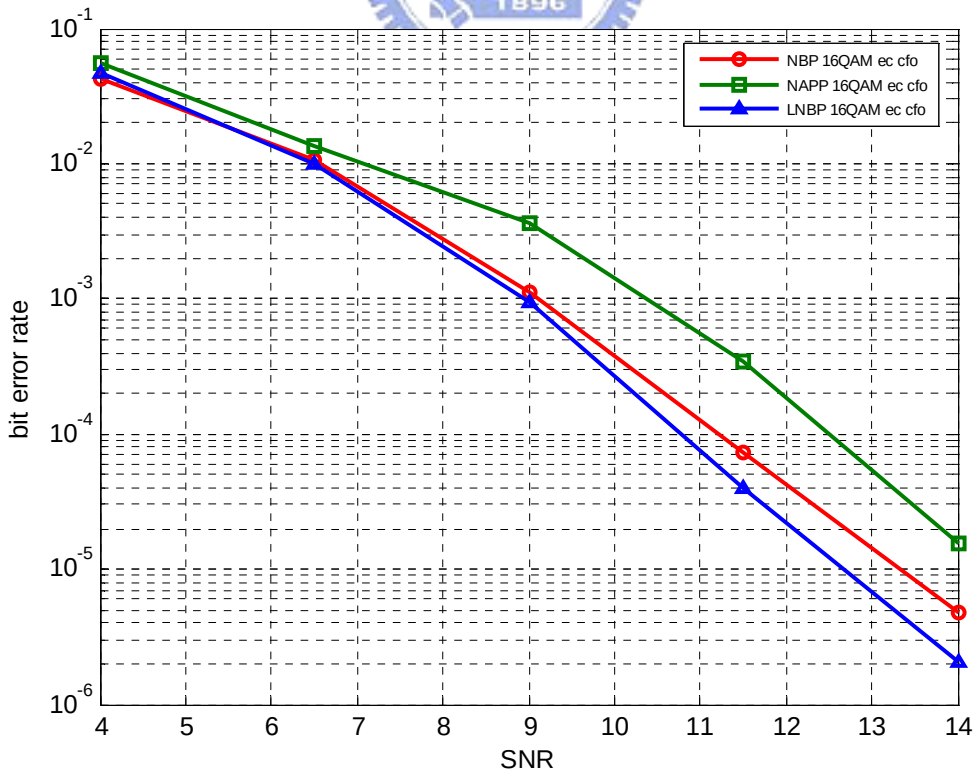


圖 3-36 802.16e 系統在 16QAM 調變、估計通道、加入 CFO 效應下模擬結果

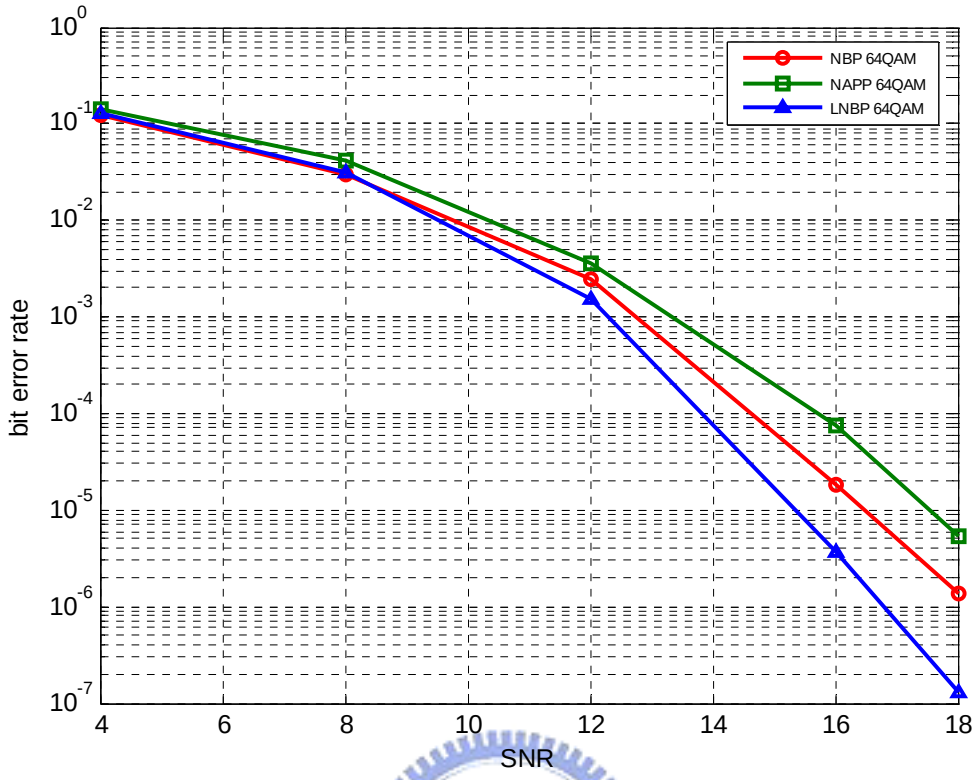


圖 3-37 802.16e 系統在 64QAM 調變、理想通道下模擬結果

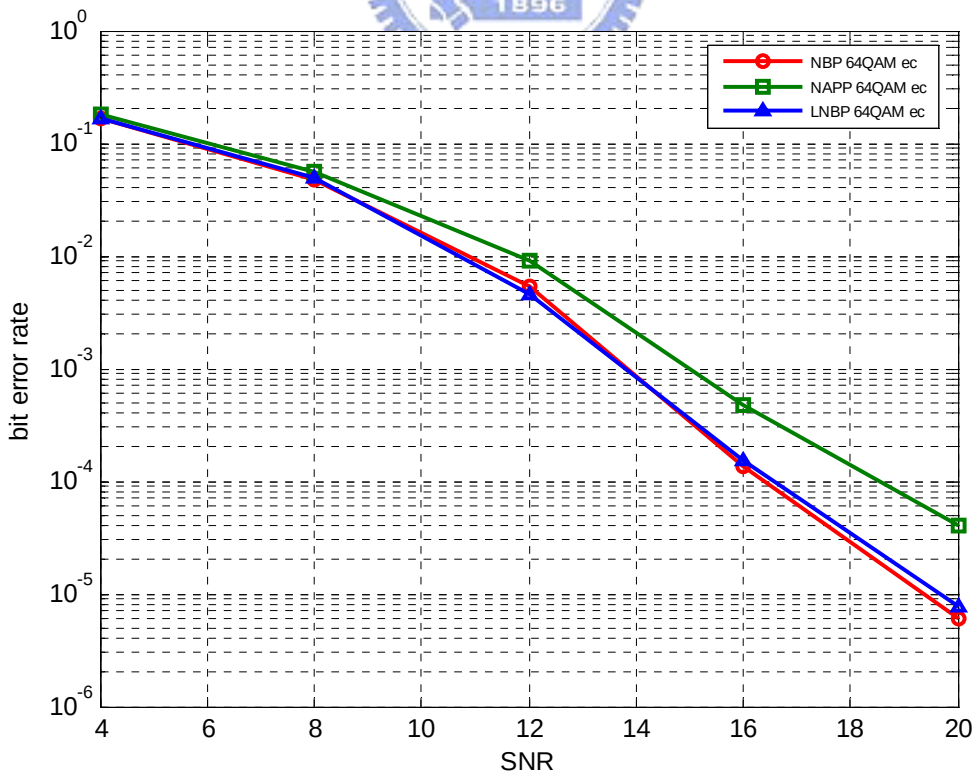


圖 3-38 802.16e 系統在 64QAM 調變、估計通道下模擬結果

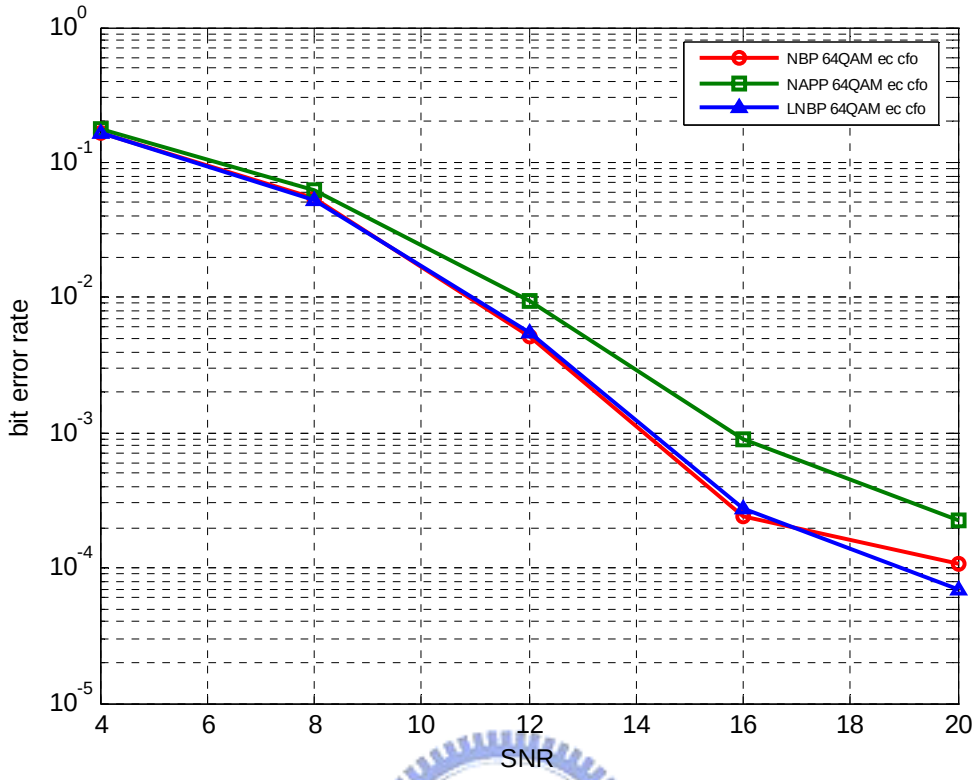


圖 3-39 802.16e 系統在 64QAM 調變、估計通道、加入 CFO 效應下模擬結果

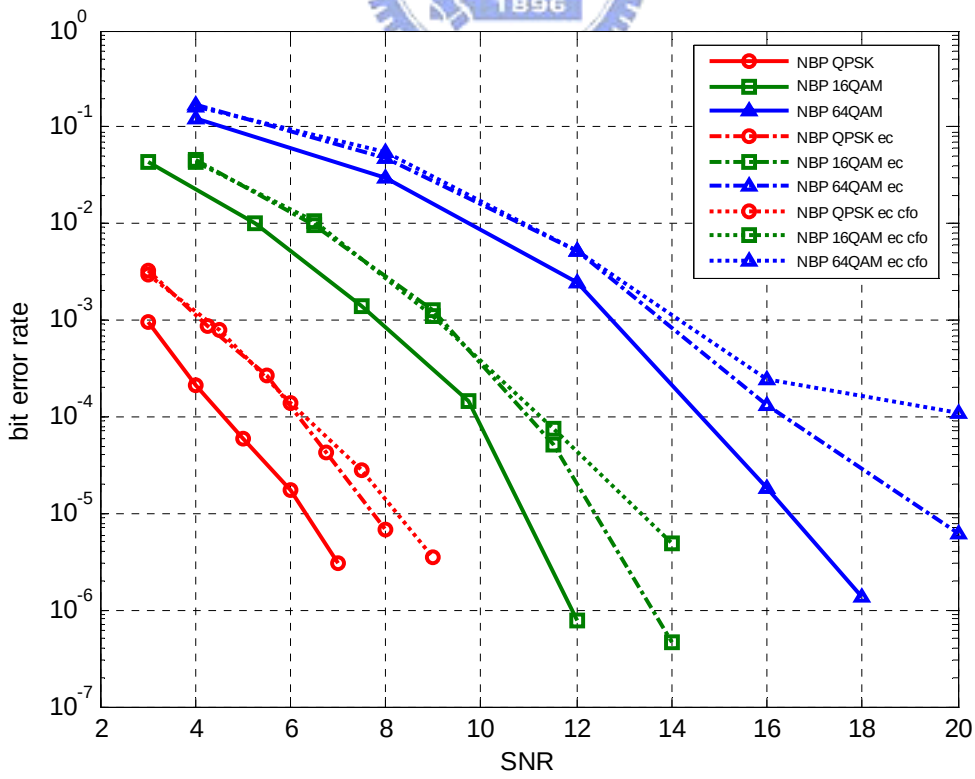


圖 3-40 NBP 解碼在 802.16e 系統下模擬結果

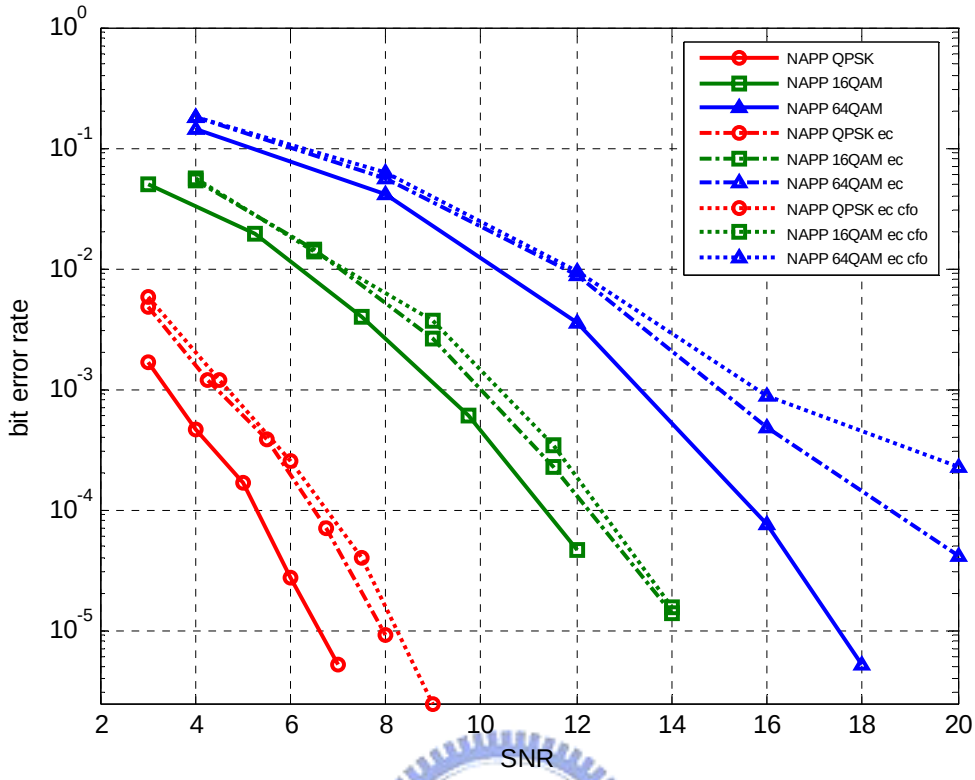


圖 3-41 NAPP 解碼在 802.16e 系統下模擬結果

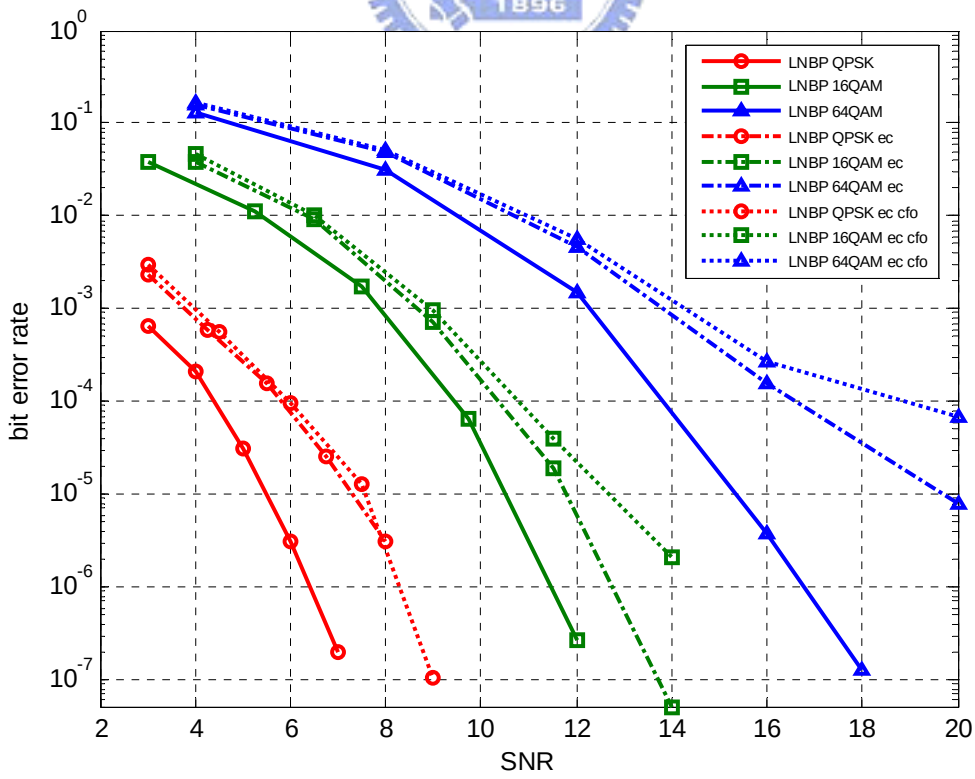


圖 3-42 LNBP 解碼在 802.16e 系統下模擬結果

第4章 最佳 MIMO 系統偵測

在上一章，我們使用 MMSE 做為 MIMO 系統之偵測器（detector），並將所偵測到的訊號透過軟性反對映來獲得軟性輸出，再將此軟性輸出送入 LDPC Codes 解碼器進行解碼。雖然使用此一方式來做為接收機的設計雖然架構簡單易於實現，然而由 (3.26) 式：

$$Z_{i,k} = \underbrace{w_i^T h_{i,k}}_{H_{eff}} s_{i,k} + \underbrace{w_i^T h_{j\neq i,k} s_{j,k} + w_i^T N_k}_{N_{eff}}$$

，我們可發現其中的等效雜訊 N_{eff} 並非白色雜訊，也就是說不同"i"之間的等化訊號 $Z_{i,k}$ 彼此會有相關性（correlation）。由於軟性反對映與 LDPC Codes 解碼器皆是在假設訊號彼此之間互為獨立的條件下對訊號進行分析的，所以在此一情況下，會使得系統的效能下降。為了解決此一問題，我們使用 MAP 偵測器，將偵測器與軟性反對映結合在一起，不必透過 MMSE 等化接收訊號後才再求取軟性輸出，而可直接由原始的接收的訊號來獲得準確性較高的軟性輸出之值，以確保系統本身的效能。

在第三章中有提到，整個 MIMO-OFDM 系統在不同的子通道之數學模型可以用 (3.2) 式：

$$Y(k) = H(k)S(k) + N(k)$$

來表示，由於 $H(k)$ 與 $N(k)$ 對不同的 k 而言是 i.i.d.隨機變數，為了表示方便，我們將 k 省略，也就是 (3.2) 式可表示為下式：

$$y = Hs + n \tag{4.1}$$

，以方便我們在數學式上的表示與推導。

4.1 MAP 位元偵測

MAP偵測器 [20]可直接由接收到的MIMO訊號來獲得軟性輸出之值，而不必透過MMSE等化所接收到的MIMO訊號，再經過軟性反對映來求得軟性輸出。藉由MAP偵測器，我們可以得到正確性較高的軟性輸出之值來提升系統效能。而使用MAP偵測器所設計之2×2 MIMO訊號接收機則如圖 4-1 所示。

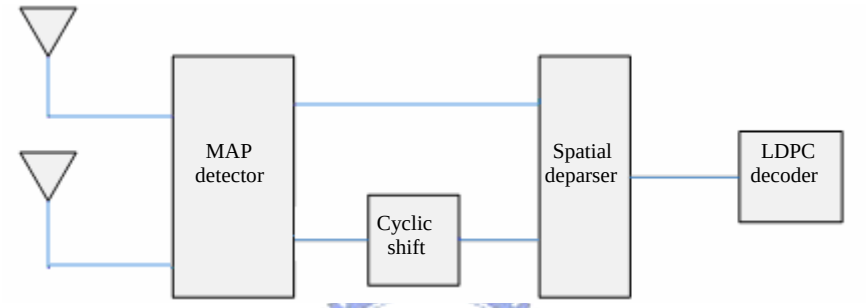


圖 4-1 2x2 MAP 偵測之接收機

若我們使用$\left(2^M\right)$QAM 調變，並以N_t根天線傳送，則$\left(4.1\right)$式中傳送的訊號向量"s"共含$M\cdot N_t$個位元，則我們可定義：

$$demap\left(s\right)=x=\left[x_0,x_1,\ldots x_i,\ldots,x_{M\cdot N_t-1}\right] \tag{4.2}$$

，其中$x_i=\pm1$，+1表示邏輯上的"1"，–1表示邏輯上的"0"，$i=0,1,\ldots,M\cdot N_t-1$。則依據$\left(2.7\right)$式 LLR 之定義，我們可得：

$$LLR\left(p\left(x_i\middle|y\right)\right)=\log\frac{P\left(x_i=+1\middle|y\right)}{P\left(x_i=-1\middle|y\right)} \tag{4.3}$$

，由貝氏定理（Baye's theorem）可知：

$$P\left(x_i=\zeta\middle|y\right)=\frac{p\left(\left(x_i=\zeta\right)\cap y\right)}{p\left(y\right)}=\frac{p\left(y\middle|x_i=\zeta\right)P\left(x_i=\zeta\right)}{\sum_{i=0}^{M\cdot N_t-1}p\left(y\middle|x_i\right)p\left(x_i\right)},\quad \zeta=\pm1 \tag{4.4}$$

，將$\left(4.4\right)$式代入$\left(4.3\right)$式中，可推得：

$$LLR\left(p\left(x_i|y\right)\right)=\log\frac{p\left(y|x_i=+1\right)P\left(x_i=+1\right)}{p\left(y|x_i=-1\right)P\left(x_i=-1\right)}$$
$$=\log\frac{P\left(x_i=+1\right)}{P\left(x_i=-1\right)}+\log\frac{p\left(y|x_i=+1\right)}{p\left(y|x_i=-1\right)} \tag{4.5}$$

，在一開始，我們假設 x_i 的事前機率是相等的，所以：

$$\log\frac{P\left(x_i=+1\right)}{P\left(x_i=-1\right)}=\log\frac{\frac{1}{2}}{\frac{1}{2}}=0 \tag{4.6}$$

，故 (4.5) 式可簡化為：

$$LLR\left(p\left(x_i|y\right)\right)=\log\frac{p\left(y|x_i=+1\right)}{p\left(y|x_i=-1\right)} \tag{4.7}$$

。由於 x 為經過交錯器處理之訊號，所以我們可假設 $x_0,x_1,\ldots,x_i,\ldots,x_{M\cdot N_t-1}$ 彼此之間互為獨立，故我們可將 (4.7) 式進一步推導為：

$$LLR\left(p\left(x_i|y\right)\right)=\log\frac{\sum_{x\in X_{i,+1}}p\left(y|x\right)}{\sum_{x\in X_{i,-1}}p\left(y|x\right)} \tag{4.8}$$

，其中 $X_{i,\zeta}$ 表示 x 中滿足 $x_i=\zeta$ 的所有集合，即：

$$X_{i,+1}=\left\{x|x_i=+1\right\},\quad X_{i,-1}=\left\{x|x_i=-1\right\} \tag{4.9}$$

。由 (4.1) 式 y 與 s 的關係式可知，$p\left(y|x\right)$ 的機率密度分佈函數可等效為：

$$p\left(y|s=map\left(x\right)\right)=\frac{\exp\left(-\frac{1}{2\sigma^2}\cdot\left\|y-H\cdot s\right\|^2\right)}{\left(2\pi\sigma^2\right)^{N_r}} \tag{4.10}$$

，其中 N_r 表示接收天線的個數。我們將 (4.10) 式代入 (4.8) 式中可求得：

$$LLR\left(p\left(x_i\middle|y\right)\right)=\log\frac{\sum_{x\in X_{i,+1}}\exp\left(-\frac{1}{2\sigma^2}\cdot\left\|y-H\cdot s\right\|^2\right)}{\sum_{x\in X_{i,-1}}\exp\left(-\frac{1}{2\sigma^2}\cdot\left\|y-H\cdot s\right\|^2\right)} \tag{4.11}$$

。為了簡化(4.11)式，我們引用“Jacobian 對數式”：

$$\begin{aligned}\text{jacln}\left(a_1,a_2\right)&:=\ln\left(e^{a_1}+e^{a_2}\right)\\&=\max\left(a_1,a_2\right)+\underbrace{\ln\left(1+e^{-\left|a_1-a_2\right|}\right)}_{r\left(\left|a_1-a_2\right|\right)}\end{aligned} \tag{4.12}$$

，由於(4.12)式中的$r\left(\left|a_1-a_2\right|\right)$極小可省略，所以 Jacobian 對數式可簡化為：

$$\text{jacln}\left(a_1,a_2\right)\approx\max\left(a_1,a_2\right) \tag{4.13}$$

，我們將(4.11)式以簡化後的 Jacobian 對數式表示則可推導為：

$$LLR\left(p\left(x_i\middle|y\right)\right)\approx\frac{1}{2}\cdot\left(\max_{x\in X_{i,+1}}\left\{-\frac{1}{\sigma^2}\cdot\left\|y-H\cdot s\right\|^2\right\}-\max_{x\in X_{i,-1}}\left\{-\frac{1}{\sigma^2}\cdot\left\|y-H\cdot s\right\|^2\right\}\right) \tag{4.14}$$

然而，即使將$LLR\left(p\left(x_i\middle|y\right)\right)$簡化至$(4.14)$式，在求取$LLR\left(p\left(x_i\middle|y\right)\right)$的計算上仍然非常的複雜，其運算複雜度會隨著接收訊號s所含的位元數呈指數成長。我們可以很容易的從(4.14)式中觀察出，在計算$LLR\left(p\left(x_i\middle|y\right)\right)$的過程中，我們必須由$2^{M\cdot N_t-1}$種可能的$s$集合中找出最大值。若我們使用 64 QAM、4 根傳送天線，則我們須由$2^{M\cdot N_t-1}\approx8.4\times10^6$種$s$可能的組合中去找出最大的$LLR\left(p\left(x_i\middle|y\right)\right)$，此一計算量實在是太大了，在硬體的設計與實現上幾乎是不可行的。為了降低運算量，我們導入List Sphere decoding演算法 [21][22][23]，這是一種以最大可能性解碼為基礎的演算法，可將MAP偵測器在計算$LLR\left(p\left(x_i\middle|y\right)\right)$時較不可能的$s$集合有效的去除，使$s$可能之集合的範圍縮小。透過List Sphere decoding演算法，MAP偵測器的運算複雜度將大幅的降低，提升硬體設計的可實現性。而我們將在下一節對List Sphere decoding演算法做詳細的討論。

4.2 List Sphere Decoding 演算法

4.2.1 搜尋半徑與搜尋中心

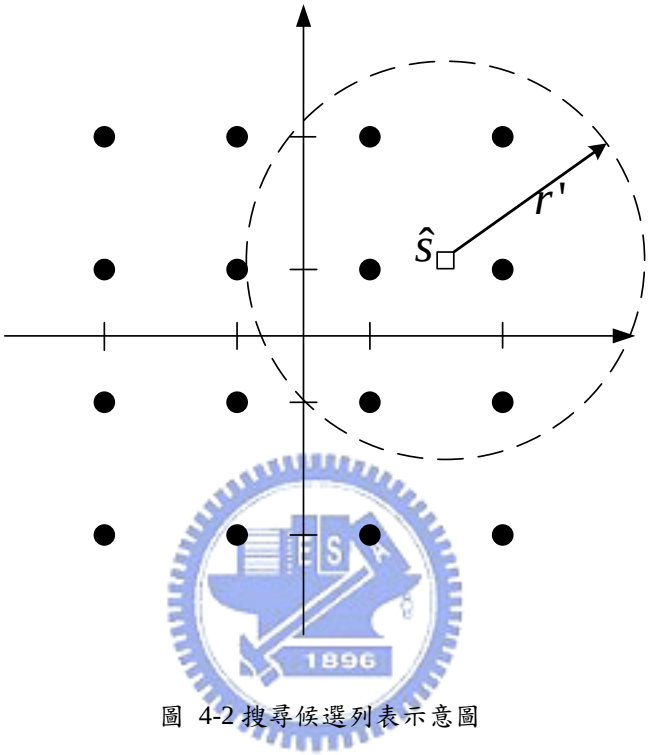


圖 4-2 搜尋候選列表示意圖

如圖 4-2 所示，List Sphere decoding 演算法的概念，是將代入$\|y-H\cdot s\|^2$所得之值過大的s集合扣除，並保留使$\|y-H\cdot s\|^2$之值較小的s集合，而所保留下的s集合我們稱之為候選列表（candidate list）"$\mathcal{L}$"。若我們定義圖 4-2 中的r'為搜尋半徑，$\hat{s}$為搜尋中心，則可將$\|y-H\cdot s\|^2$式加上搜尋半徑的限制並修正為：

$$\|y-H\cdot s\|^2 \le (r')^2 \tag{4.15}$$

，搜尋中心$\hat{s}$則可定義為接收訊號y之最小平方近似解（least-squares approximation solution）：

$$\hat{s}=\left(H^H H\right)^{-1} H^H y \tag{4.16}$$

，或是接收訊號 y 之最小均方差估測，其中 α 為 SNR 之倒數：

$$\hat{s} = \left[H^H H + \alpha I_{N_t \times N_t} \right]^{-1} H^H y \tag{4.17}$$

我們將 $H\hat{s} - H\hat{s}$ 加入 (4.15) 式中可得：

$$\begin{aligned} \left\| y - H \cdot s \right\|^2 &= \left\| y - H\hat{s} + H\hat{s} - Hs \right\|^2 \\ &= \left\| y - H\hat{s} + H\left(\hat{s} - s\right) \right\|^2 \le \left(r'\right)^2 \end{aligned} \tag{4.18}$$

，若搜尋中心 $\hat{s}$ 為 y 之最小平方近似解，(4.18) 式可推導為：

$$\begin{aligned} \left\| y - H \cdot s \right\|^2 &= \left(s - \hat{s}\right)^H H^H H \left(s - \hat{s}\right) \\ &\quad + y^H \left(I + H\left(H^H H\right)^{-1} H^H \right) y \le \left(r'\right)^2 \end{aligned} \tag{4.19}$$

令：

$$r_1^2 = \left(r'\right)^2 - y^H \left(I + H\left(H^H H\right)^{-1} H^H \right) y \tag{4.20}$$

，則 (4.18) 式可簡化為：

$$\left(s - \hat{s}\right)^H H^H H \left(s - \hat{s}\right) \le r_1^2 \tag{4.21}$$

。由於：

$$\left\| y - H \cdot s \right\|^2 = \left\| n \right\|^2 \sim \sigma^2 \cdot \chi_{2N_r}^2 \tag{4.22}$$

，(4.22) 式中之 σ^2 為雜訊 n 的實數部份之變異數；而 $\chi_{2N_r}^2$ 為有 $2N_r$ 階自由度的 chi-square 隨機變數，且其變異數為 $2N_r\sigma^2$。則我們可定義搜尋半徑“r_1”為：

$$r_1^2 = 2N_r K\sigma^2 - y^H \left(I + H\left(H^H H\right)^{-1} H^H \right) y \tag{4.23}$$

，其中 $K \ge 1$ 為一任意變數。同樣地，若搜尋中心 $\hat{s}$ 為 y 之最小均方差估測，(4.18) 式亦可推導為：

$$\begin{aligned} \left\| y - H \cdot s \right\|^2 &= \left(s - \hat{s}\right)^H H^H H \left(s - \hat{s}\right) \\ &\quad + y^H \left(I + H\left(H^H H + \alpha I_{N_t \times N_t}\right)^{-1} H^H \right) y \le \left(r'\right)^2 \end{aligned} \tag{4.24}$$

令：

$$r_2^2 = (r')^2 - y^H \left(I + H \left(H^H H + \alpha I_{N_t \times N_t} \right)^{-1} H^H \right) y \tag{4.25}$$

，則(4.18)式同樣可簡化為：

$$(s - \hat{s})^H H^H H (s - \hat{s}) \le r_2^2 \tag{4.26}$$

，而搜尋半徑“r_2”可定義為：

$$r_2^2 = 2N_r K\sigma^2 - y^H \left(I + H \left(H^H H + \alpha I_{N_t \times N_t} \right)^{-1} H^H \right) y \tag{4.27}$$

。由於(4.21)式與(4.26)式兩式之形式相同，故之後的推導皆適用於兩式上，為了表示方便，我們定義：

$$(s - \hat{s})^H H^H H (s - \hat{s}) \le r^2 \tag{4.28}$$

，以方便我們的推導。

4.2.2 實數 List Sphere decoding 演算法

由於 List Sphere decoding 演算法在複數的處理上相當複雜，為了簡化運算，我們將 y 、s 、$\hat{s}$ 、H 轉為實數等效矩陣 y_γ 、s_γ 、$\hat{s}_\gamma$ 與 H_γ ：

$$y_\gamma = \begin{bmatrix} \mathrm{Re}(y)^T & \mathrm{Im}(y)^T \end{bmatrix}^T \tag{4.29}$$

$$s_\gamma = \begin{bmatrix} \mathrm{Re}(s)^T & \mathrm{Im}(s)^T \end{bmatrix}^T \tag{4.30}$$

$$\hat{s}_\gamma = \begin{bmatrix} \mathrm{Re}(\hat{s})^T & \mathrm{Im}(\hat{s})^T \end{bmatrix}^T \tag{4.31}$$

$$H_\gamma = \begin{bmatrix} \mathrm{Re}(H) & -\mathrm{Im}(H) \\ \mathrm{Im}(H) & \mathrm{Re}(H) \end{bmatrix} \tag{4.32}$$

，其中 $\mathrm{Re}(a)$ 表示取矩陣 a 之實部所產生之矩陣，$\mathrm{Im}(a)$ 表示取矩陣 a 之虛部所產生之矩陣，故 y_γ 為 $2N_r \times 1$ 的矩陣，s_γ 與 $\hat{s}_\gamma$ 皆為 $2N_t \times 1$ 的矩陣，H_γ 則為

$2N_r \times 2N_t$ 的矩陣。可發現將 s 、 $\hat{s}_\gamma$ 與 H_γ 代入 (4.15) 式與 (4.28) 式中仍成立：

$$\left\| y_\gamma - H_\gamma \cdot s_\gamma \right\|^2 \le (r')^2 \tag{4.33}$$

$$\left(s - \hat{s}_\gamma\right)^H H_\gamma{}^H H_\gamma \left(s - \hat{s}_\gamma\right) \le r^2 \tag{4.34}$$

接著我們使用 Cholesky factorization，求得：

$$U^H U = H_\gamma{}^H H_\gamma \tag{4.35}$$

，其中 U 為一 $2N_t \times 2N_t$ 的上三角矩陣（upper triangular matrix），並定義 U 之元素為 $u_{i,j}$， $i \le j = 1,2,\ldots,2N_t$ ，且其對角線上的元素 $u_{ii} > 0$ ， $s_{\gamma i}$ 和 $\hat{s}_{\gamma i}$ 分別表示向量 s_γ 和向量 $\hat{s}_\gamma$ 中的第 i 個元素，則我們可將 (4.31) 式推導為：

$$\begin{aligned} &\left(s_\gamma - \hat{s}_\gamma\right)^H U^H U \left(s_\gamma - \hat{s}_\gamma\right) \\ &= \sum_{i=1}^{2N_t} u_{i,i}^2 \left[s_{\gamma i} - \hat{s}_{\gamma i} + \sum_{j=i+1}^{2N_t} \frac{u_{i,j}}{u_{i,i}} \left(s_{\gamma i} - \hat{s}_{\gamma i}\right) \right]^2 \le r^2 \end{aligned} \tag{4.36}$$

。List Sphere decoding 演算法在求得候選列表的過程，是如同樹狀圖（tree diagrams）一層接著一層來進行搜尋，一開始，我們先從 $i = 2N_t$ 開始搜尋，並忽略 $i = 1,2,\ldots,2N_t - 1$ 項所產生的影響，依據上述 (4.36) 式可推導為：

$$u_{2N_t,2N_t}^2 \left(s_{\gamma 2N_t} - \hat{s}_{\gamma 2N_t}\right)^2 \le r^2 \tag{4.37}$$

，我們可由 (4.37) 式求得 $s_{\gamma 2N_t}$ 的上、下限：

$$\left\lceil \hat{s}_{\gamma 2N_t} - \frac{r}{u_{2N_t,2N_t}} \right\rceil \le s_{\gamma 2N_t} \le \left\lfloor \hat{s}_{\gamma 2N_t} + \frac{r}{u_{2N_t,2N_t}} \right\rfloor \tag{4.38}$$

，其中 $\lceil a \rceil$ 表示取 (2^M) QAM 星狀圖之實數軸上大於等於 a 且最接近 a 的對映

點，而$\lfloor a \rfloor$則表示取$\left(2^M\right)$QAM 星狀圖之實數軸上小於等於a且最接近a的對映點。$s_{\gamma 2N_t}$的上、下限決定後，我們可知道$s_{\gamma 2N_t}$所有可能的選項，任選其中的一個選項，並令$i=2N_t-1$忽略$i=1,2,\ldots,2N_t-2$項所產生的影響，可再將(4.36)式推導為：

$$u_{2N_t-1,2N_t-1}^2\left[s_{\gamma 2N_t-1}-\left(\hat{s}_{\gamma 2N_t-1}-\frac{u_{2N_t-1,2N_t}}{u_{2N_t,2N_t}}\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)\right)\right]^2 + u_{2N_t,2N_t}^2\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)^2 \le r^2 \tag{4.39}$$

，而$s_{\gamma 2N_t-1}$的下限為：

$$s_{\gamma 2N_t-1} \ge \left\lceil \left(\hat{s}_{\gamma 2N_t-1}-\frac{u_{2N_t-1,2N_t}}{u_{2N_t,2N_t}}\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)\right)-\left(\frac{r^2-u_{2N_t,2N_t}^2\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)^2}{u_{2N_t-1,2N_t-1}^2}\right)^{\frac{1}{2}} \right\rceil \tag{4.40}$$

上限為：

$$s_{\gamma 2N_t-1} \le \left\lfloor \left(\hat{s}_{\gamma 2N_t-1}-\frac{u_{2N_t-1,2N_t}}{u_{2N_t,2N_t}}\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)\right)+\left(\frac{r^2-u_{2N_t,2N_t}^2\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)^2}{u_{2N_t-1,2N_t-1}^2}\right)^{\frac{1}{2}} \right\rfloor \tag{4.41}$$

。同樣地，令$i=2N_t-2$忽略$i=1,2,\ldots,2N_t-3$項所產生的影響，選取$s_{\gamma 2N_r-1}$所有可能的其中一個選項，並與之前選定的$s_{\gamma 2N_t}$代入(4.36)式可得：

$$u_{2N_t-2,2N_t-2}^2\left[s_{\gamma 2N_t-2}-\left(\hat{s}_{\gamma 2N_t-2}-\frac{u_{2N_t-2,2N_t-1}}{u_{2N_t-2,2N_t-2}}\left(s_{\gamma 2N_t-1}-\hat{s}_{\gamma 2N_t-1}\right)\right.\right.$$
$$\left.\left.-\frac{u_{2N_t-2,2N_t}}{u_{2N_t-2,2N_t-2}}\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)\right)\right]^2$$
$$+u_{2N_t-1,2N_t-1}^2\left[s_{\gamma 2N_t-1}-\left(\hat{s}_{\gamma 2N_t-1}-\frac{u_{2N_t-1,2N_t}}{u_{2N_t,2N_t}}\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)\right)\right]^2 \quad (4.42)$$
$$+u_{2N_t,2N_t}^2\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)^2\le r^2$$

，而 $s_{\gamma 2N_t-2}$ 的下限為：

$$s_{\gamma 2N_t-2}\ge$$
$$\left\lceil\left(\hat{s}_{\gamma 2N_t-2}-\frac{u_{2N_t-2,2N_t-1}}{u_{2N_t-2,2N_t-2}}\left(s_{\gamma 2N_t-1}-\hat{s}_{\gamma 2N_t-1}\right)-\frac{u_{2N_t-2,2N_t}}{u_{2N_t-2,2N_t-2}}\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)\right)\right.$$
$$-\left\{\left\{r^2-u_{2N_t-1,2N_t-1}^2\left[s_{\gamma 2N_t-1}-\left(\hat{s}_{\gamma 2N_t-1}-\frac{u_{2N_t-1,2N_t}}{u_{2N_t,2N_t}}\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)\right)\right]^2\right.\right. \quad (4.43)$$
$$\left.\left.\left.-u_{2N_t,2N_t}^2\left(s_{\gamma 2N_t}-\hat{s}_{\gamma 2N_t}\right)^2\right\}\middle/u_{2N_t-2,2N_t-2}^2\right\}^{\frac{1}{2}}\right\rceil$$

，$s_{\gamma 2N_t-2}$ 的上限為：

$$s_{\gamma 2N_t-2} \le \left\lfloor \left(\hat{s}_{\gamma 2N_t-2} - \frac{u_{2N_t-2,2N_t-1}}{u_{2N_t-2,2N_t-2}}\left(s_{\gamma 2N_t-1} - \hat{s}_{\gamma 2N_t-1}\right) - \frac{u_{2N_t-2,2N_t}}{u_{2N_t-2,2N_t-2}}\left(s_{\gamma 2N_t} - \hat{s}_{\gamma 2N_t}\right) \right) + \left\{ \left\{ r^2 - u_{2N_t-1,2N_t-1}^2 \left[s_{\gamma 2N_t-1} - \left(\hat{s}_{\gamma 2N_t-1} - \frac{u_{2N_t-1,2N_t}}{u_{2N_t,2N_t}}\left(s_{\gamma 2N_t} - \hat{s}_{\gamma 2N_t}\right) \right) \right]^2 - u_{2N_t,2N_t}^2 \left(s_{\gamma 2N_t} - \hat{s}_{\gamma 2N_t}\right)^2 \right\} \Big/ u_{2N_t-2,2N_t-2}^2 \right\}^{\frac{1}{2}} \right\rfloor \quad (4.44)$$

。在求取 $s_{\gamma 2N_t-3}$ 的上、下限亦同上述的步驟，令 $i = 2N_t - 3$ 忽略 $i = 1,2,\ldots,2N_t - 4$ 項，並將選取的 $s_{\gamma 2N_t}$ 、 $s_{\gamma 2N_t-1}$ 及 $s_{\gamma 2N_t-2}$ 代入 (4.36) 式即可，然而上述計算過程過於繁雜不利於硬體的設計，但我們觀察 (4.38) 式、(4.40) 式、(4.41) 式、(4.43) 式與 (4.44) 式，發現可將求取上、下限之運算式整理成一適合執行遞迴運算的通式。首先令：

$$\left(\hat{s}_\gamma\right)_{2N_t|2N_t+1} = \hat{s}_{\gamma 2N_t}, \quad \left(\hat{s}_\gamma\right)_{i|i+1} = \hat{s}_{\gamma i} - \sum_{j=i+1}^{2N_t} \frac{u_{i,j}}{u_{i,i}}\left(s_{\gamma j} - \hat{s}_{\gamma j}\right) \quad (4.45)$$

$$r''^2_{2N_t} = r^2, \quad r''^2_i = r''^2_{i+1} - u_{i+1,i+1}^2 \left(s_{\gamma i+1} - \left(\hat{s}_\gamma\right)_{i+1|i+2}\right)^2 \quad (4.46)$$

$$z_i = \frac{r''_i}{u_{i,i}} \quad (4.47)$$

，則上、下限之運算式可改寫為：

$$\left\lceil \left(\hat{s}_\gamma\right)_{i|i+1} - z_i \right\rceil \le s_{\gamma i} \le \left\lfloor \left(\hat{s}_\gamma\right)_{i|i+1} + z_i \right\rfloor \quad (4.48)$$

，如此，我們可以很容易的以疊代和遞迴的方式來計算 $s_{\gamma i}$ 的上、下限。

4.2.3 實數 List Sphere decoding 演算法運算流程

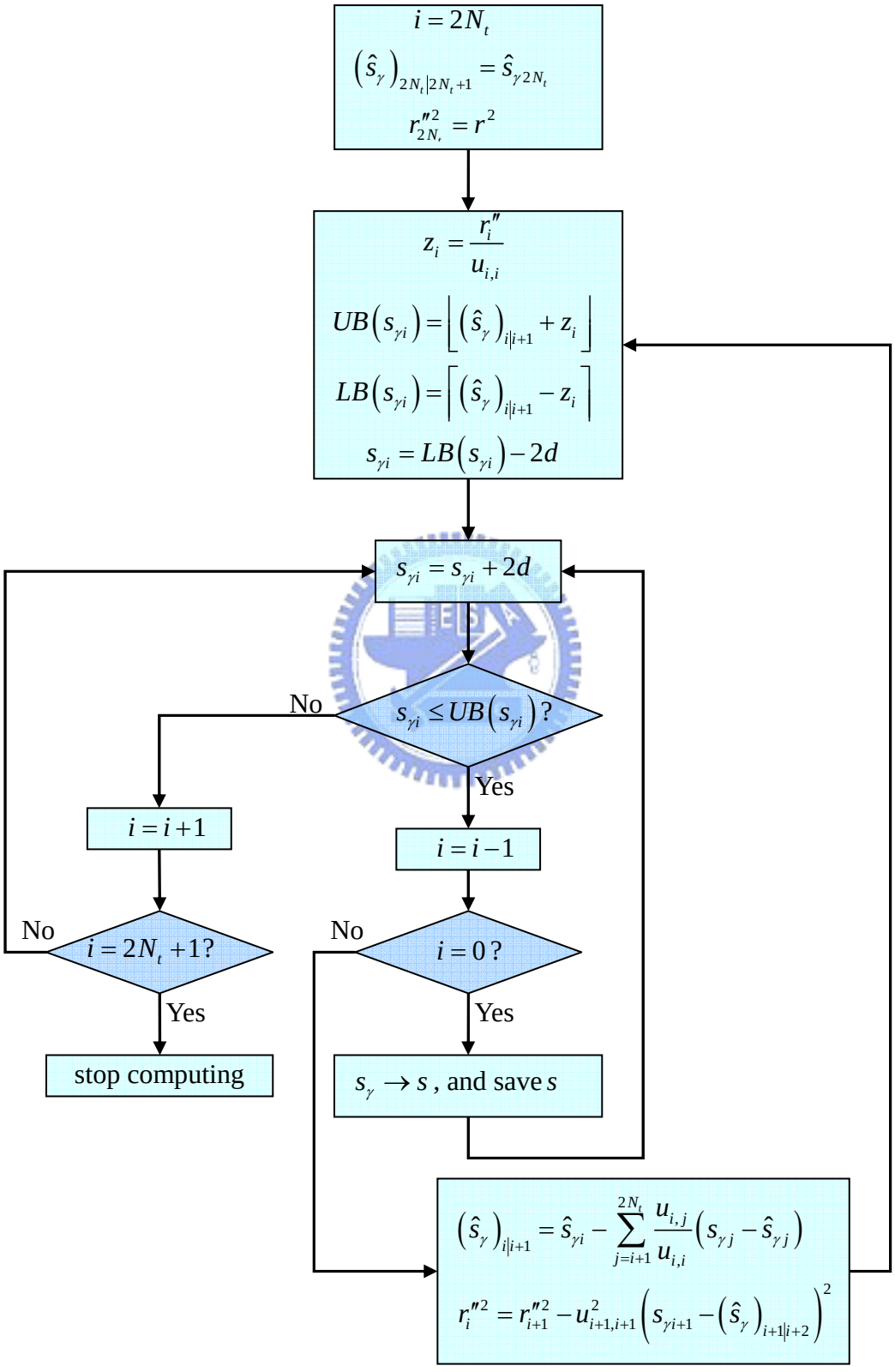


圖 4-3 實數 List Sphere decoding 演算法運算流程圖

如圖 4-3 所示：

第一步：初始化，令：

$$i = 2N_t \tag{4.49}$$

$$\left(\hat{s}_\gamma\right)_{2N_t|2N_t+1} = \hat{s}_{\gamma 2N_t} \tag{4.50}$$

$$r''^2_{2N_t} = r^2 \tag{4.51}$$

第二步：計算上、下限，令：

$$z_i = \frac{r''_i}{u_{i,i}} \tag{4.52}$$

，則上限$UB\left(s_{\gamma i}\right)$、下限$LB\left(s_{\gamma i}\right)$為：

$$UB\left(s_{\gamma i}\right) = \left\lfloor \left(\hat{s}_\gamma\right)_{i|i+1} + z_i \right\rfloor \tag{4.53}$$

$$LB\left(s_{\gamma i}\right) = \left\lceil \left(\hat{s}_\gamma\right)_{i|i+1} - z_i \right\rceil \tag{4.54}$$

$$s_{\gamma i} = LB\left(s_{\gamma i}\right) - 2d \tag{4.55}$$

，其中$2d$為$\left(2^M\right)$QAM 星狀圖上對映點間的最小距離。

第三步：增加$s_{\gamma i}$：

$$s_{\gamma i} = s_{\gamma i} + 2d \tag{4.56}$$

若重新計算所得之$s_{\gamma i} > UB\left(s_{\gamma i}\right)$，則進行第四步運算；若重新計算所得之

$s_{\gamma i} \le UB\left(s_{\gamma i}\right)$，則執行第五步運算。

第四步：增加i：

$$i = i + 1 \tag{4.57}$$

，若計算所得之$i = 2N_t + 1$，則結束 List Sphere decoding 演算法計算；否則，跳回第三步運算。

第五步：減少i：

$$i = i - 1 \tag{4.58}$$

，若計算所得之 $i=0$，則執行第六步的運作；否則：

$$\left(\hat{s}_{\gamma}\right)_{i|i+1}=\hat{s}_{\gamma i}-\sum_{j=i+1}^{2N_t}\frac{u_{i,j}}{u_{i,i}}\left(s_{\gamma j}-\hat{s}_{\gamma j}\right) \tag{4.59}$$

$$r_i''^2=r_{i+1}''^2-u_{i+1,i+1}^2\left(s_{\gamma i+1}-\left(\hat{s}_{\gamma}\right)_{i+1|i+2}\right)^2 \tag{4.60}$$

，並回到第二步運算。

第六步：將所求得之 s_{γ} 轉換回 s，再將 s 儲存於候選列表 $\mathcal{L}$ 中，並跳回第三步運算。

4.2.4 實數 List Sphere decoding 演算法修正

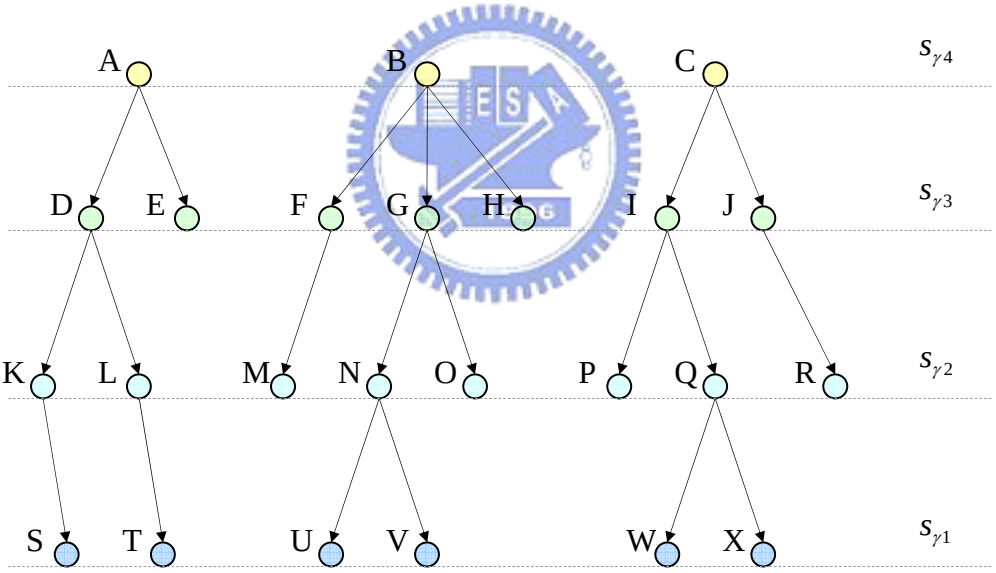


圖 4-4 $N_t=2$ 實數 List Sphere decoding 演算法搜尋圖

如圖 4-4 所示，實數 List Sphere decoding 演算法的運算過程，是如同樹狀圖一層接著一層來進行搜尋，所以搜尋路徑（path）的多寡（這裡我們定義由一個節點連結到另一個節點為一個路徑），會決定 List Sphere decoding 演算法的運算複雜度。由 4.2.2 節的討論，我們知道實數 List Sphere decoding 演算法是由 $i=2N_t$

開始搜尋的，由於$i=2N_t$為樹狀圖的最上層，$i=1$則為樹狀圖的最下層，故只有在$i=1$時有連結之節點的候選（如圖 4-4 之 A-D-K-S、B-G-N-U、B-G-N-V、C-I-Q-W、C-I-Q-X），才會儲存於候選列表$\mathcal{L}$中，而其它在$i=1$時沒有連結之節點者（如圖 4-4 之 A-E、B-F-M、B-G-O、B-H、C-I-P、C-J-R），會產生多餘的路徑（如 A-E、B-F、F-M、B-G、G-O、B-H、C-I、I-P、C-J、J-R），若能減少多餘路徑的搜尋，便能降低 List Sphere decoding 演算法的計算量。為此，吾人提出兩種修正方法，可在較上層的運算中，便將多數在$i=1$時沒有連結之節點者排除，減少多餘路徑的搜尋。

首先將(4.45)式代入(4.46)式可得：

$$r_i''^2 = r_{i+1}''^2 - u_{i+1,i+1}^2\left(s_{\gamma i+1} - \left(\hat{s}_{\gamma i+1} - \sum_{j=i+2}^{2N_t}\frac{u_{i+1,j}}{u_{i+1,i}}\left(s_{\gamma j} - \hat{s}_{\gamma j}\right)\right)\right)^2 \tag{4.61}$$

，其中$r_{i'}''$為搜尋半徑r扣除$i=2N_t \sim i=i'+1$層所產生之影響的等效搜尋半徑，而較小的等效搜尋半徑$r_{i'}''$，其所搜尋到的可能的$s_{\gamma i'}$數目亦會較少，故只要能讓等效搜尋半徑$r_{i'}''$在較上層的運算中迅速的縮小，即可將多數在$i=1$時沒有連結之節點者排除。要使等效搜尋半徑$r_{i'}''$在較上層的運算中就變小，我們可以對搜尋中心$\hat{s}_\gamma$內其元素之絕對值進行排序，其值較小者置於$\hat{s}_\gamma^T$的前面，其值較大者則置於$\hat{s}_\gamma^T$的後面，而實數等效通道矩陣H_γ之行向量，亦要隨$\hat{s}_\gamma$元素置換的順序做相同的置換，舉例如下，假設有一$N_t=2$之系統，則可定義：

$$\hat{s}_\gamma = \left[\hat{s}_{\gamma 1}, \hat{s}_{\gamma 2}, \hat{s}_{\gamma 3}, \hat{s}_{\gamma 4}\right]^T \tag{4.62}$$

$$H_\gamma = \left[H_{\gamma 1}, H_{\gamma 2}, H_{\gamma 3}, H_{\gamma 4}\right] \tag{4.63}$$

，若$\hat{s}_\gamma$經過排序後為：

$$\hat{s}_\gamma' = \left[\hat{s}_{\gamma 4}, \hat{s}_{\gamma 1}, \hat{s}_{\gamma 3}, \hat{s}_{\gamma 2}\right]^T, \quad \left|\hat{s}_{\gamma 4}\right| \le \left|\hat{s}_{\gamma 1}\right| \le \left|\hat{s}_{\gamma 3}\right| \le \left|\hat{s}_{\gamma 2}\right| \tag{4.64}$$

，則實數等效通道矩陣H_γ之行向量亦須進行相同的置換：

$$H_\gamma' = \left[H_{\gamma 4}, H_{\gamma 1}, H_{\gamma 3}, H_{\gamma 2} \right] \tag{4.65}$$

，即可將 $\hat{s}_\gamma'$ 與 H_γ' 代入實數 List Sphere decoding 演算法，最後將求得的 s_γ' 進行反置換為 s_γ ：

$$s_\gamma = \left[s_{\gamma 2}', s_{\gamma 4}', s_{\gamma 3}', s_{\gamma 1}' \right] \tag{4.66}$$

，再儲存於候選列表 $\mathcal{L}$ 內。由於只要搜尋半徑 r 不變，搜尋到的候選列表 $\mathcal{L}$ 亦不變，所以經過排序後的 $\hat{s}_\gamma'$ 與 H_γ' 並不會改變候選列表 $\mathcal{L}$ 。

另外，我們亦可對實數等效通道矩陣 H_γ 之行向量的模數（Norm）進行排序，其值較小者置於 H_γ 的前面，其值較大者則置於 H_γ 的後面，同樣地，搜尋中心 $\hat{s}_\gamma$ 亦須隨 H_γ 之行向量置換的順序做相同的置換，依照 (4.62) 與 (4.63) 式的例子；若 H_γ 經過排序後為：

$$H_\gamma'' = \left[H_{\gamma 3}, H_{\gamma 1}, H_{\gamma 2}, H_{\gamma 4} \right], \quad \left\| H_{\gamma 3} \right\| \le \left\| H_{\gamma 1} \right\| \le \left\| H_{\gamma 2} \right\| \le \left\| H_{\gamma 4} \right\| \tag{4.67}$$

，則搜尋中心 $\hat{s}_\gamma$ 之元素亦須進行相同的置換：

$$\hat{s}_\gamma'' = \left[\hat{s}_{\gamma 3}, \hat{s}_{\gamma 1}, \hat{s}_{\gamma 2}, \hat{s}_{\gamma 4} \right]^T \tag{4.68}$$

，再將 $\hat{s}_\gamma''$ 與 H_γ'' 代入實數 List Sphere decoding 演算法，最後把求得的 $\hat{s}_\gamma''$ 進行反置換為 s_γ ：

$$s_\gamma = \left[s_{\gamma 2}'', s_{\gamma 3}'', s_{\gamma 1}'', s_{\gamma 4}'' \right] \tag{4.69}$$

，再儲存於候選列表 $\mathcal{L}$ 內。我們將上述兩種經過修正後的實數 List Sphere decoding 演算法和原來演算法的運算複雜度與候選列表數進行測試，並將結果表示於圖 4-5 到圖 4-8。

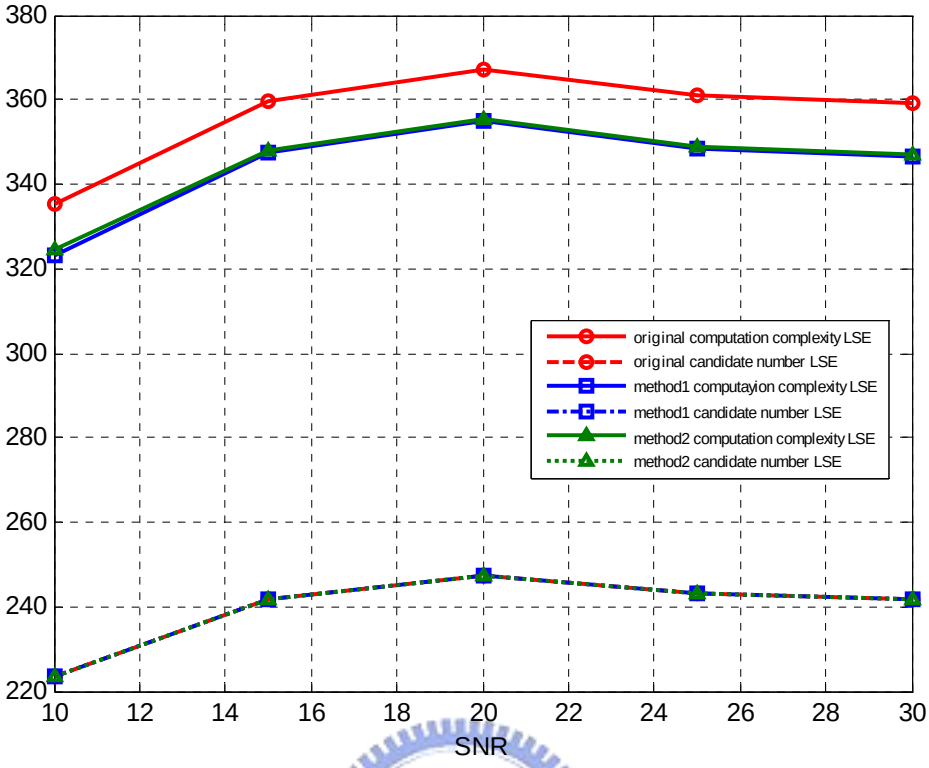


圖 4-5 LSE 搜尋中心在 64QAM 2x2 通道下測試結果

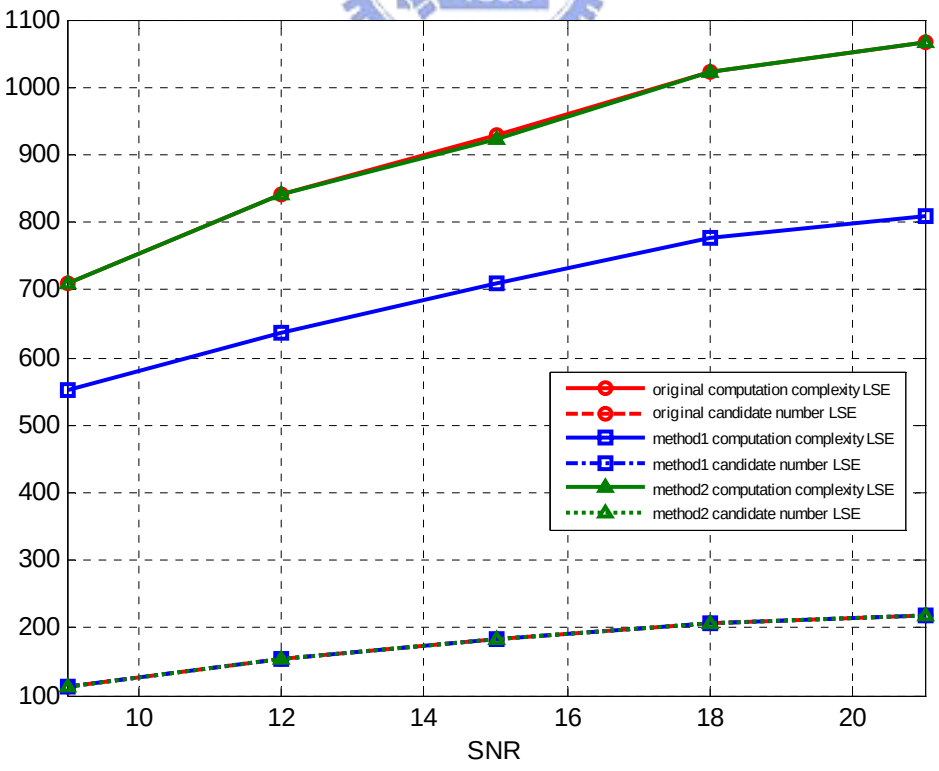


圖 4-6 LSE 搜尋中心在 16QAM 4x4 通道下測試結果

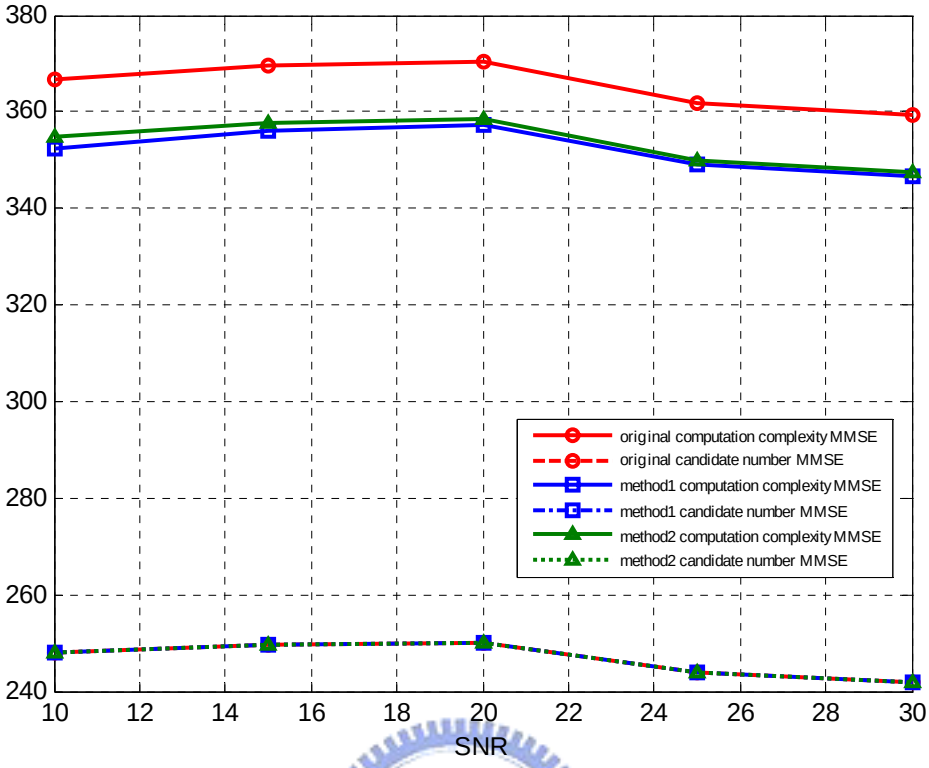


圖 4-7 MMSE 搜尋中心在 64QAM 2x2 通道下測試結果

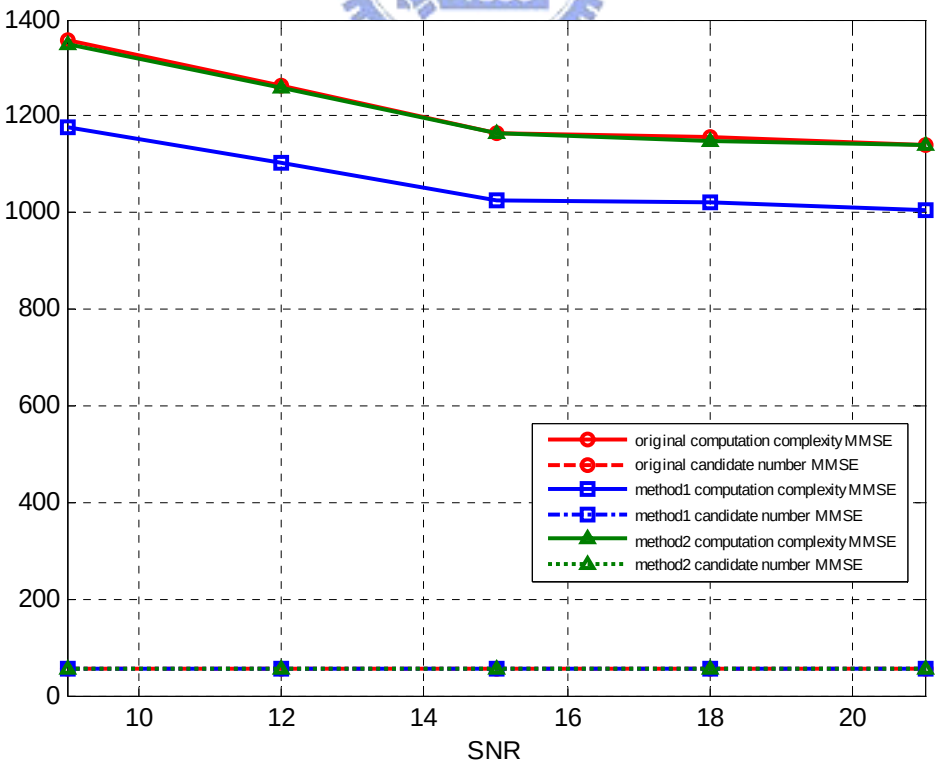


圖 4-8 MMSE 搜尋中心在 16QAM 4x4 通道下測試結果

圖 4-5 到圖 4-8 皆是在單一 tap 數 Rayleigh 分佈的 MIMO 通道上進行測試，其中在 64 QAM、2×2 通道下我們所使用的搜尋半徑為 0.4，而在 16 QAM、4×4 通道下我們所使用的搜尋半徑則為 0.3；圖中的 original 表示原來的實數 List Sphere decoding 演算法，method1 表示對搜尋中心 $\hat{s}_\gamma$ 之絕對值排序的實數 List Sphere decoding 演算法，method2 則表示對實數等效通道矩陣 H_γ 之行向量的模數排序的實數 List Sphere decoding 演算法；而圖 4-5 和圖 4-6 中的 LSE 是表示以接收訊號 y 之最小平方近似解為搜尋中心，圖 4-7 和圖 4-8 中的 MMSE 則表示以接收訊號 y 之最小均方差估測為搜尋中心；computation complexity 線上之值表示平均的路徑個數，candidate number 線上之值則為平均的候選列表之候選個數。

由圖 4-5 到圖 4-8 可確定，使用對搜尋中心 $\hat{s}_\gamma$ 之絕對值排序的實數 List Sphere decoding 演算法，不論是在 64 QAM 2×2 通道環境下，還是在 16 QAM 4×4 通道環境下，皆有最低的平均路徑個數，在 16 QAM 4×4 通道環境下甚至可以減少快 200 次路徑的搜尋。使用對實數等效通道矩陣 H_γ 之行向量的模數排序的實數 List Sphere decoding 演算法，在 64 QAM 2×2 通道環境下，其平均路徑個數雖只比對搜尋中心 $\hat{s}_\gamma$ 之絕對值排序的演算法略大一點，但在 16 QAM 4×4 通道環境下，其平均路徑個數卻與原來未修正的演算法幾乎相同，沒有太大的改進，這應該是因為在 4×4 的通道環境下，實數等效通道矩陣 H_γ 會被延伸為 8×8 的矩陣，其矩陣的行向量有 8 個元素之多，而隨著實數等效通道矩陣之行向量內之元素的增加，不同行向量所得模數之值會越接近，在此情形下對 H_γ 之行向量做排序已無太大的意義，因為行向量彼此之間模數差距不大，其對等效搜尋半徑 r_i'' 的影響亦已降低。但不論是使用 method1 修正的演算法，還是使用 method2 修正的演算法，其候選列表內平均的候選個數皆與原來未修正的演算法相同，不會對系統的錯誤率有影響。

4.3 Max-Log-MAP List Sphere 偵測器

透過 List Sphere decoding 演算法，我們可將較不可能的 s 集合去除，而獲得可能之 s 集合的候選列表 $\mathcal{L}$，MAP 偵測器在計算 $LLR\left(p\left(x_i|y\right)\right)$ 時只需在 $\mathcal{L}$ 內找出最大值即可，故 (4.14) 式可簡化為：

$$LLR\left(p\left(x_i|y\right)\right)\approx\frac{1}{2}\left(\max_{x\in\mathcal{L}\cap X_{i,+1}}\left\{-\frac{1}{\sigma^2}\cdot\left\|y-H\cdot s\right\|^2\right\}-\max_{x\in\mathcal{L}\cap X_{i,-1}}\left\{-\frac{1}{\sigma^2}\cdot\left\|y-H\cdot s\right\|^2\right\}\right) \tag{4.70}$$

，若候選列表 $\mathcal{L}$ 內並不存在 X_i 為 $+1$ 或 X_i 為 -1 的向量訊號 s，即：

$$\mathcal{L}\cap X_{i,+1}=[\ ]\quad\text{or}\quad\mathcal{L}\cap X_{i,-1}=[\ ] \tag{4.71}$$

，通常表示此傳送之訊號內 X_i 為 $+1$ 或 X_i 為 -1 的機率極低，此時會給 $\left\|y-H\cdot s\right\|^2$ 一個負的極大值或正的極大值，以確保MAP偵測器與LDPC Codes解碼器的正常運作。而此Max-Log-MAP List Sphere偵測器 [24]在 2×2 通道環境下之架構如圖 4-9 所示。

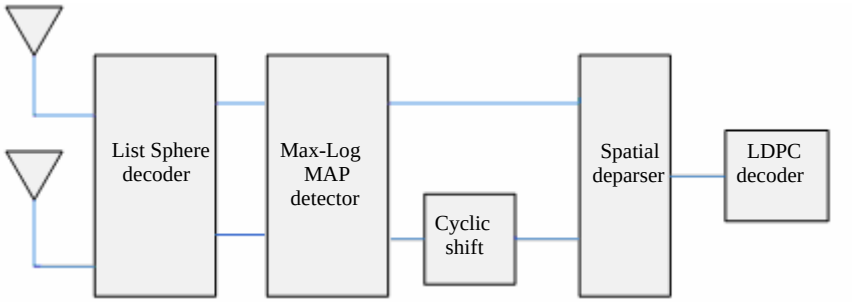


圖 4-9 2x2 之 Max-Log MAP List Sphere 偵測器

4.4 802.11n 系統模擬結果

4.4.1 的模擬結果中，虛線表示者的系統模擬環境除了將 MMSE 偵測及軟性反對映，改以 Max-Log-MAP List Sphere 偵測器來取代對接收訊號進行偵測與軟性輸出的計算外，其餘模擬環境皆與 3.8.1 相同，我們把使用 Max-Log-MAP List Sphere 偵測器的系統以“MLSD”來表示。另外，我們將 3.8.1 使用 MMSE 偵測及軟性反對映的模擬結果，以實線標示在相同的圖上，並以“MMSE”來表示。

圖 4-10 到圖 4-15 都是使用 Sum-product 演算法，在估計通道並假設完美同步下的模擬結果，其中我們設定最大遞迴次數為50。圖 4-16 到圖 4-33 則與 3.8.1 的圖 3-13 到圖 3-30 模擬環境相同，都加上了相同的非理想效應，且最大遞迴次數皆為15；其中圖 4-16 到圖 4-21 使用 Normalized BP-based 演算法解碼，圖 4-22 到圖 4-27 使用 Normalized APP-based 演算法解碼，4-28 到圖 4-33 使用 Layered Normalized BP-based 演算法解碼。

由圖 4-10 到圖 4-33 可發現，使用 Max-Log-MAP List Sphere 偵測器系統的效能很明顯的優於使用 MMSE 偵測及軟性反對映的系統。在封包錯誤率（packet error rate）同樣為10^{-1}的情況下，4×4通道幾乎都有5至6 dB 以上的改進，MCS28在 B 通道下甚至會有10 dB 的增益。另外使用 MCS28，Max-Log-MAP List Sphere 偵測器的系統，不管在何種通道下，都比使用 MCS27，MMSE 偵測及軟性反對映的系統有更低的封包錯誤率。而2×2通道也幾乎都有3 dB 以上的改進，在 B 通道下，使用 MCS15，Max-Log-MAP List Sphere 偵測器的系統，也比使用 MCS13，MMSE 偵測及軟性反對映的系統有更好的表現。

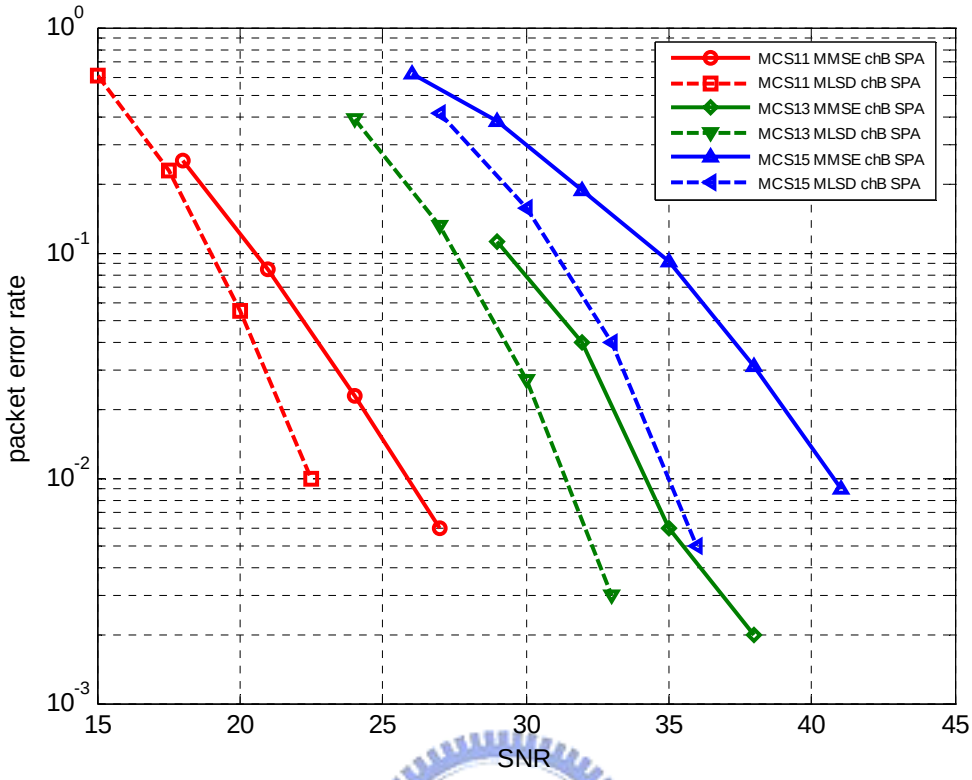


圖 4-10 SPA 解碼在 2x2 通道 B 下模擬結果

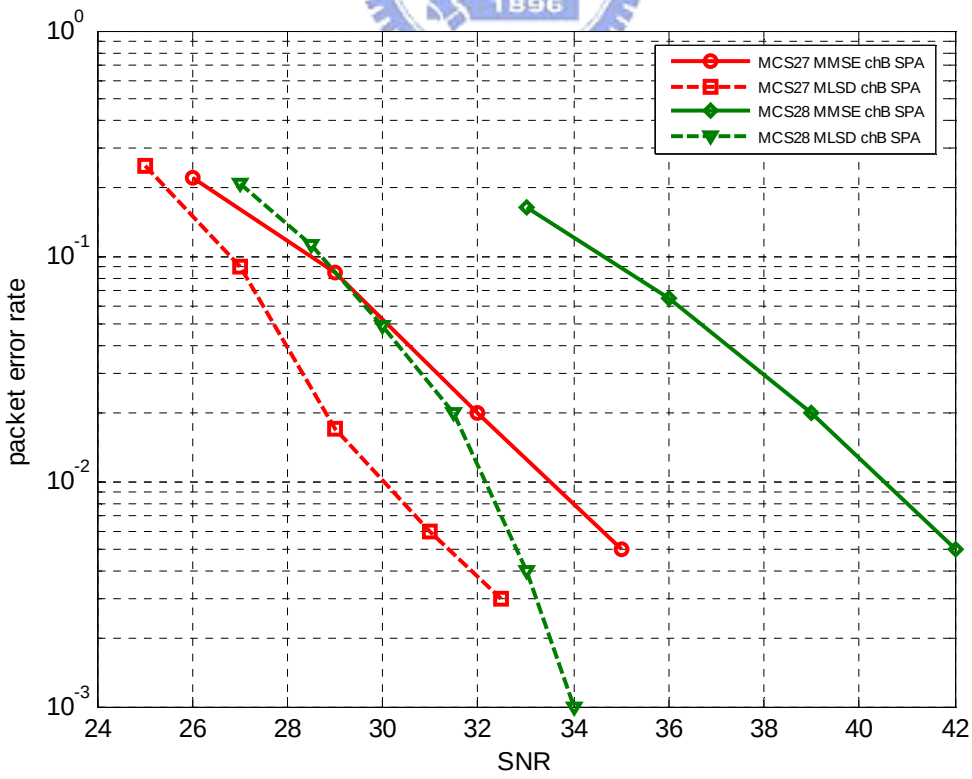


圖 4-11 SPA 解碼在 4x4 通道 B 下模擬結果

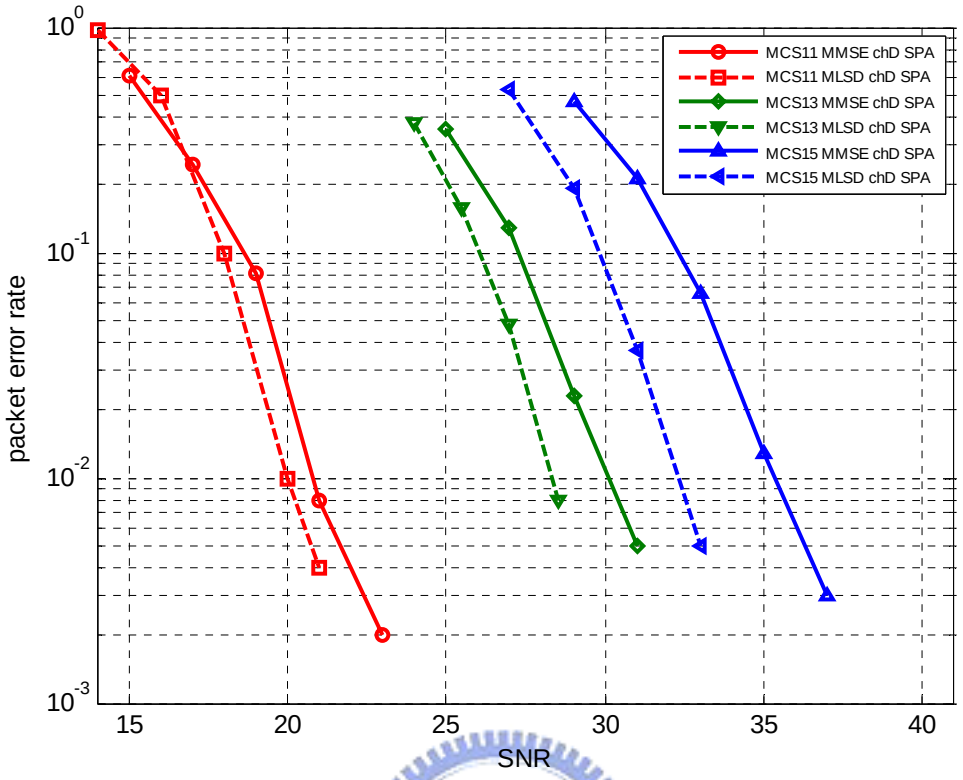


圖 4-12 SPA 解碼在 2x2 通道 D 下模擬結果

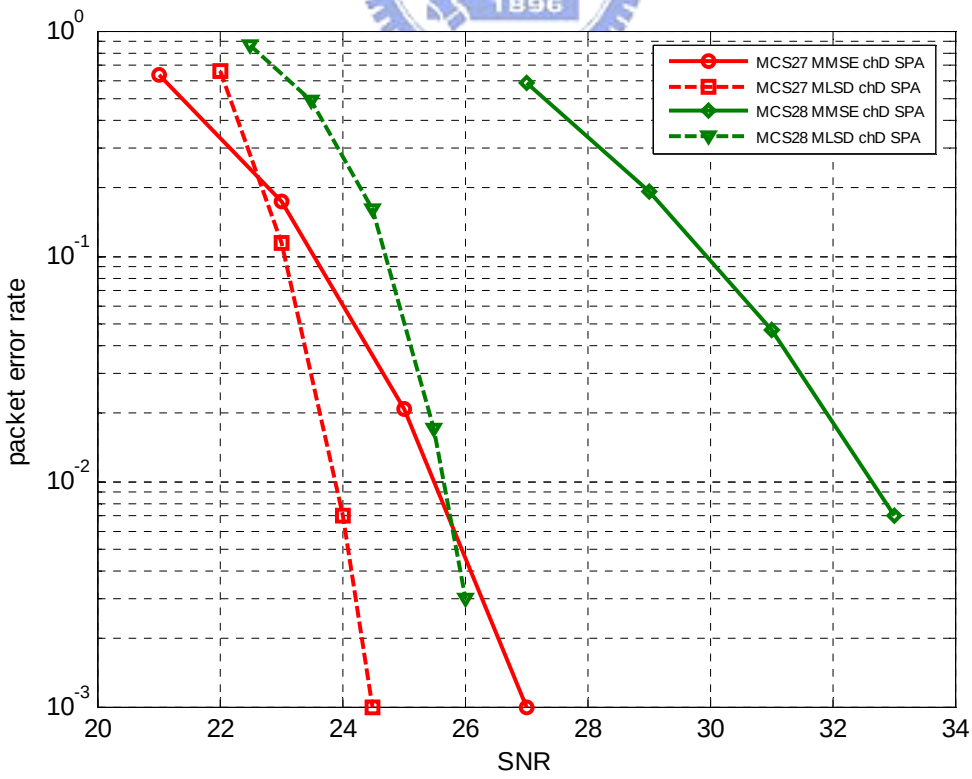


圖 4-13 SPA 解碼在 4x4 通道 D 下模擬結果

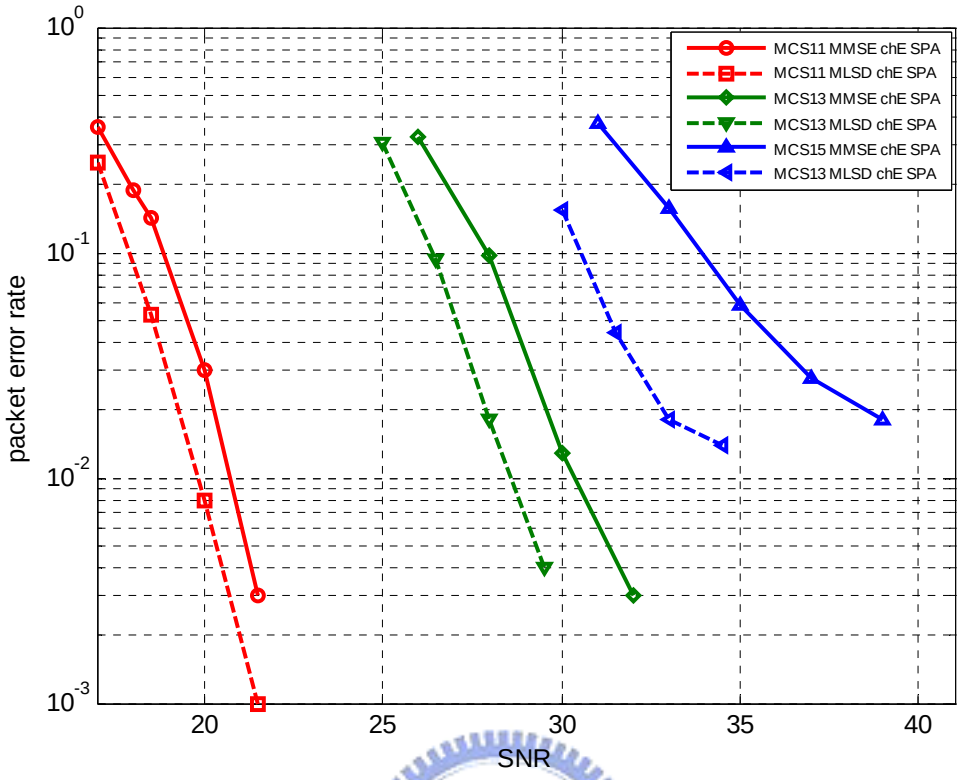


圖 4-14 SPA 解碼在 2x2 通道 E 下模擬結果

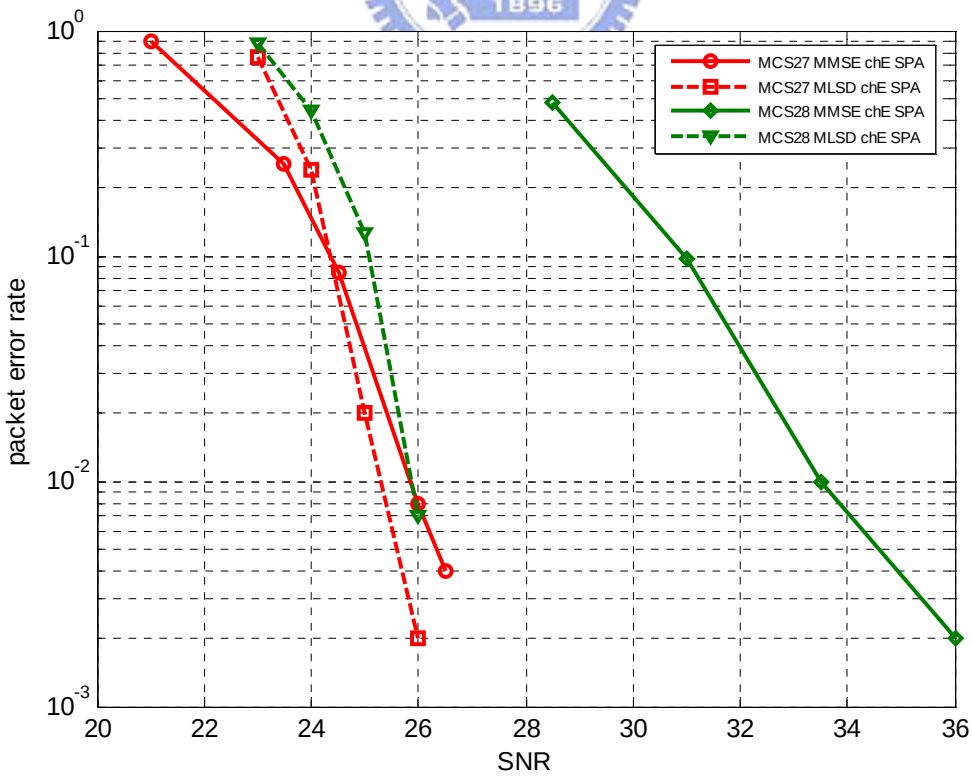


圖 4-15 SPA 解碼在 4x4 通道 E 下模擬結果

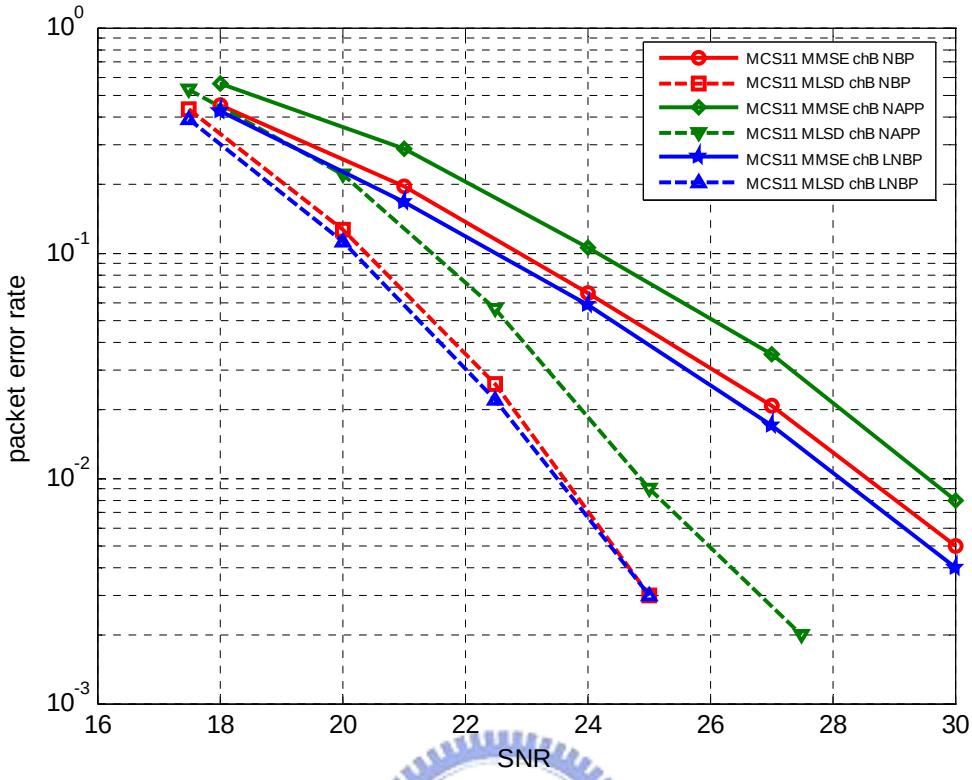


圖 4-16 MCS11 在 2x2 通道 B 下模擬結果

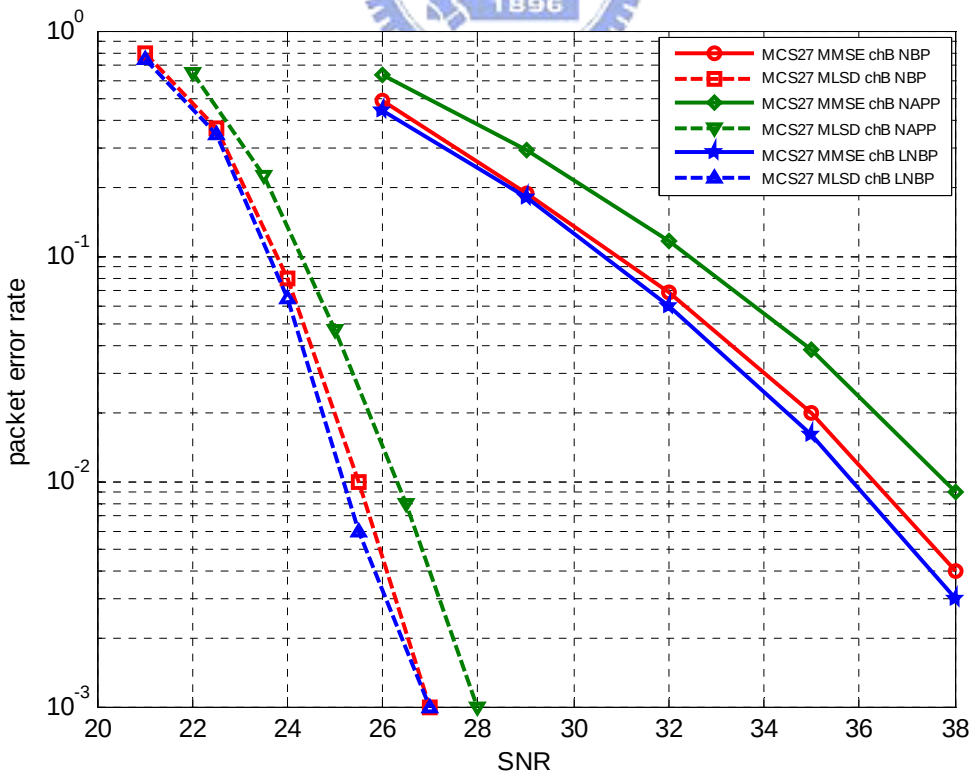


圖 4-17 MCS27 在 4x4 通道 B 下模擬結果

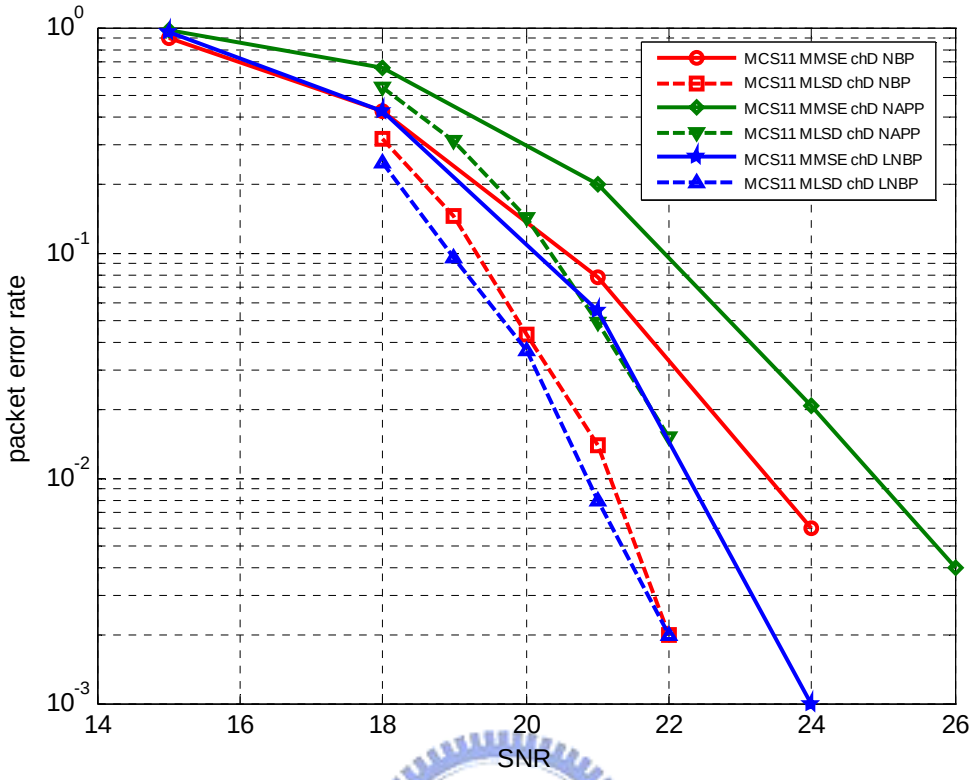


圖 4-18 MCS11 在 2x2 通道 D 下模擬結果

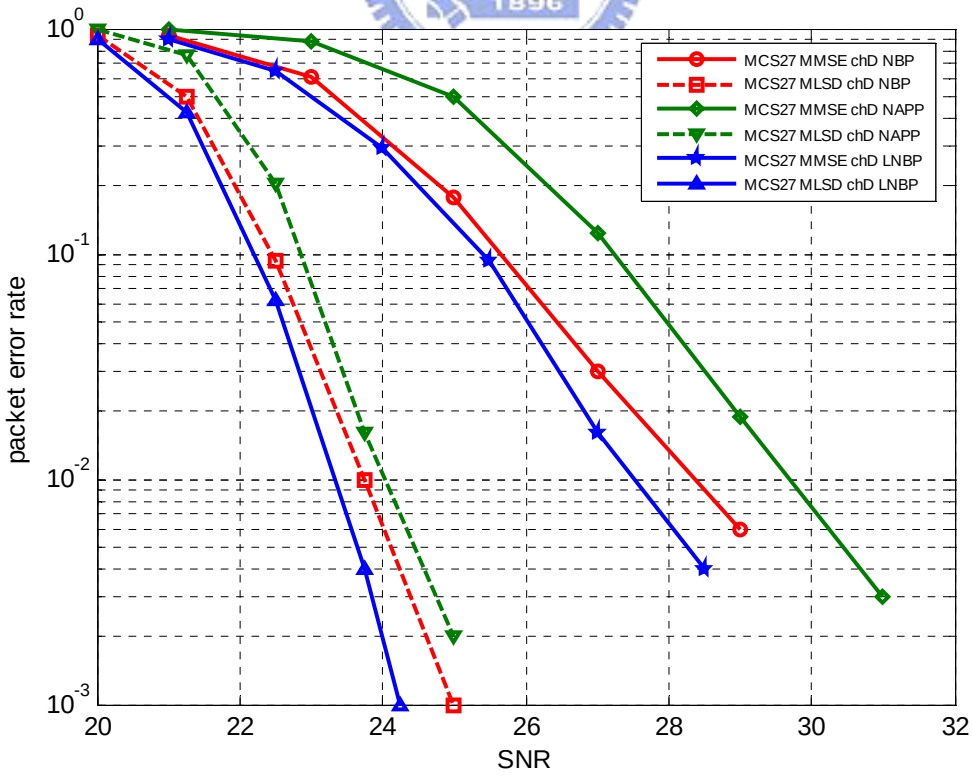


圖 4-19 MCS27 在 4x4 通道 D 下模擬結果

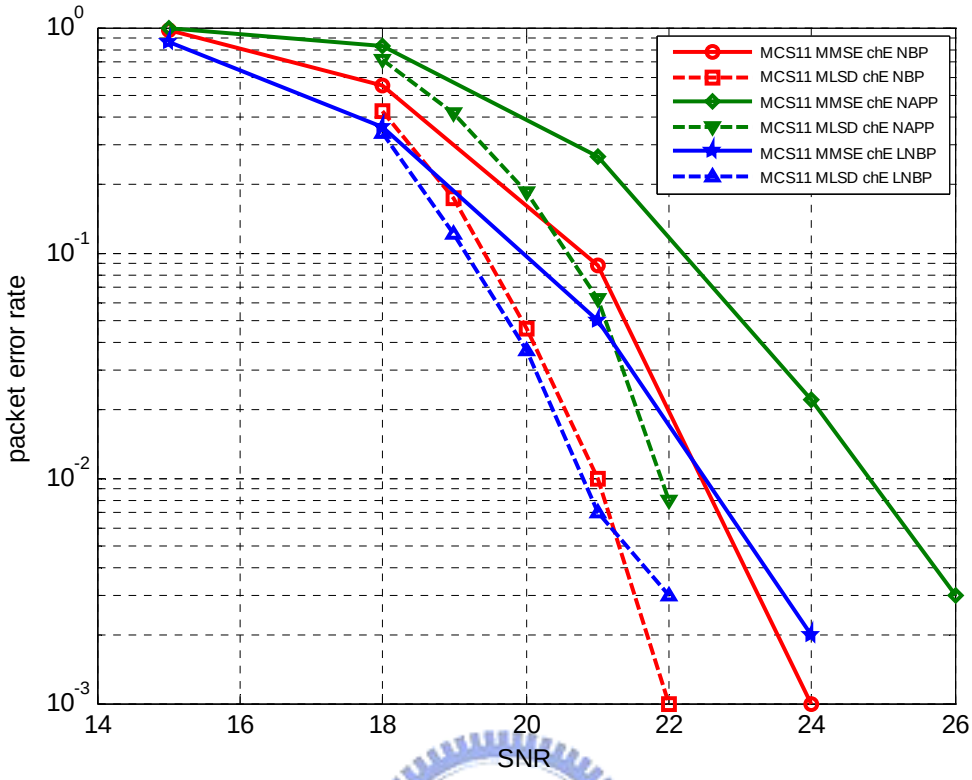


圖 4-20 MCS11 在 2x2 通道 E 下模擬結果

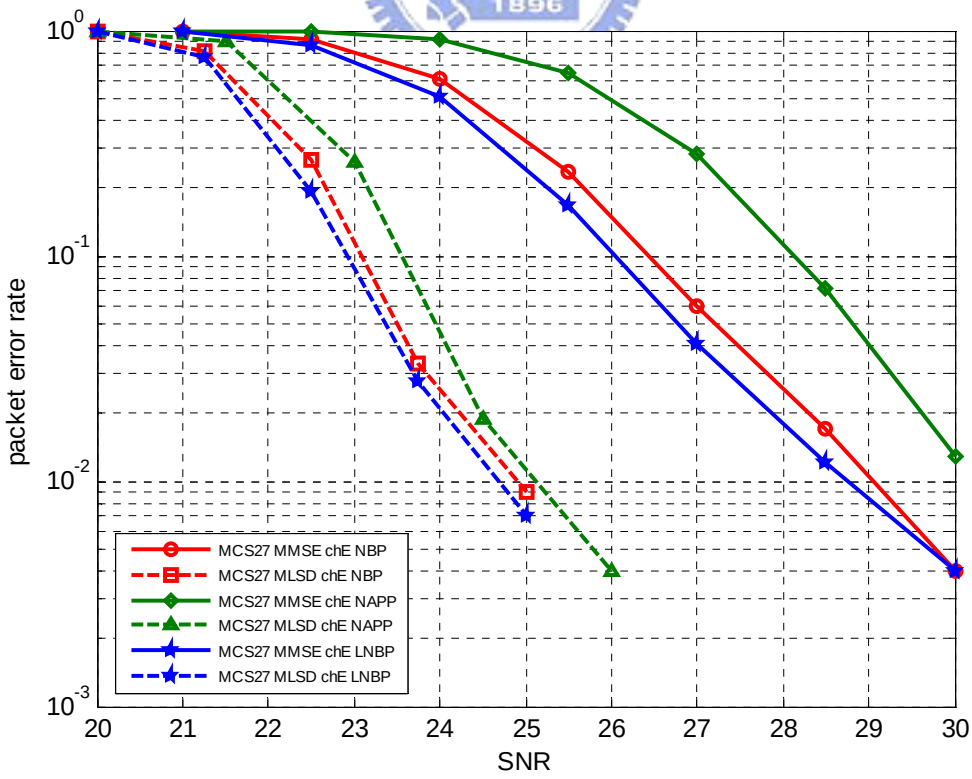


圖 4-21 MCS27 在 4x4 通道 E 下模擬結果

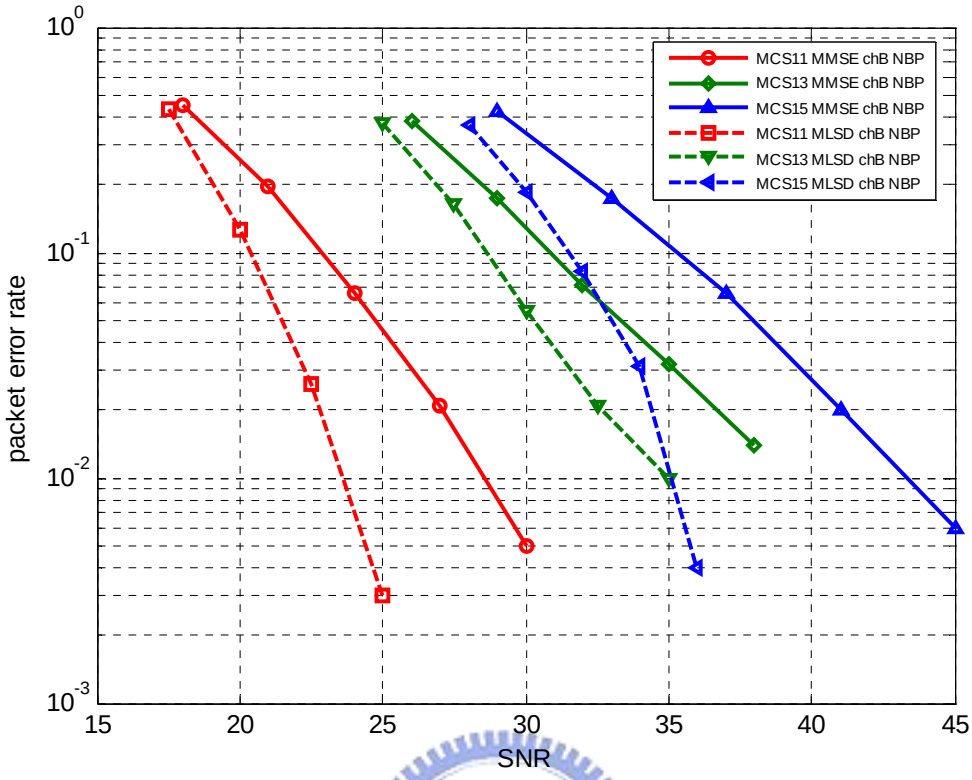


圖 4-22 NBP 解碼在 2x2 通道 B 下模擬結果

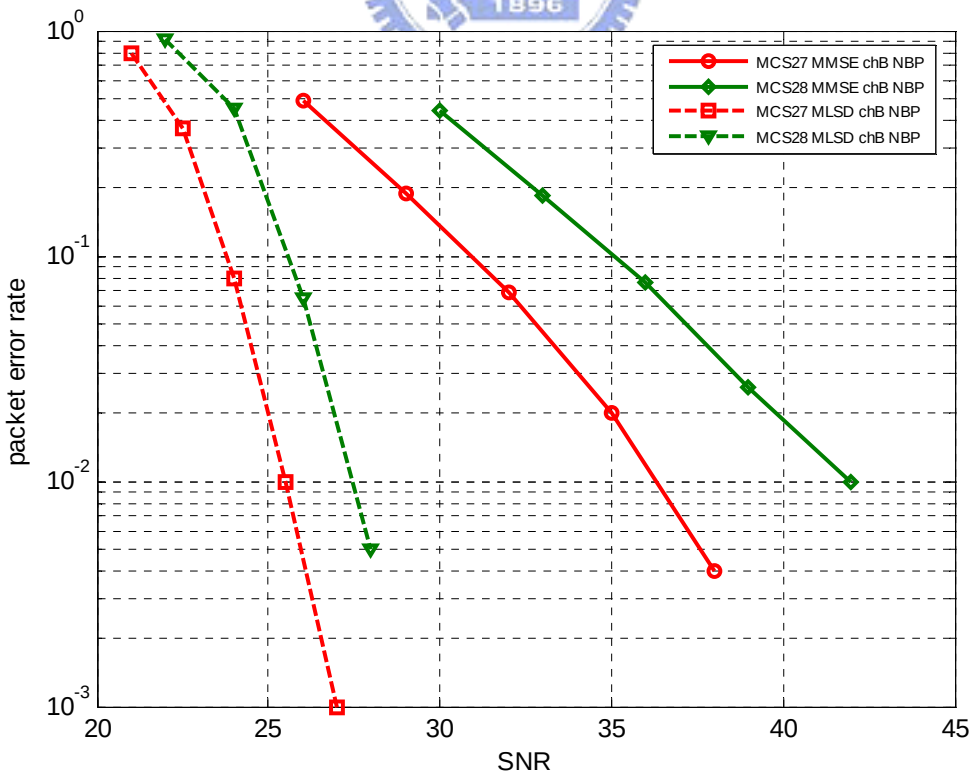


圖 4-23 NBP 解碼在 4x4 通道 B 下模擬結果

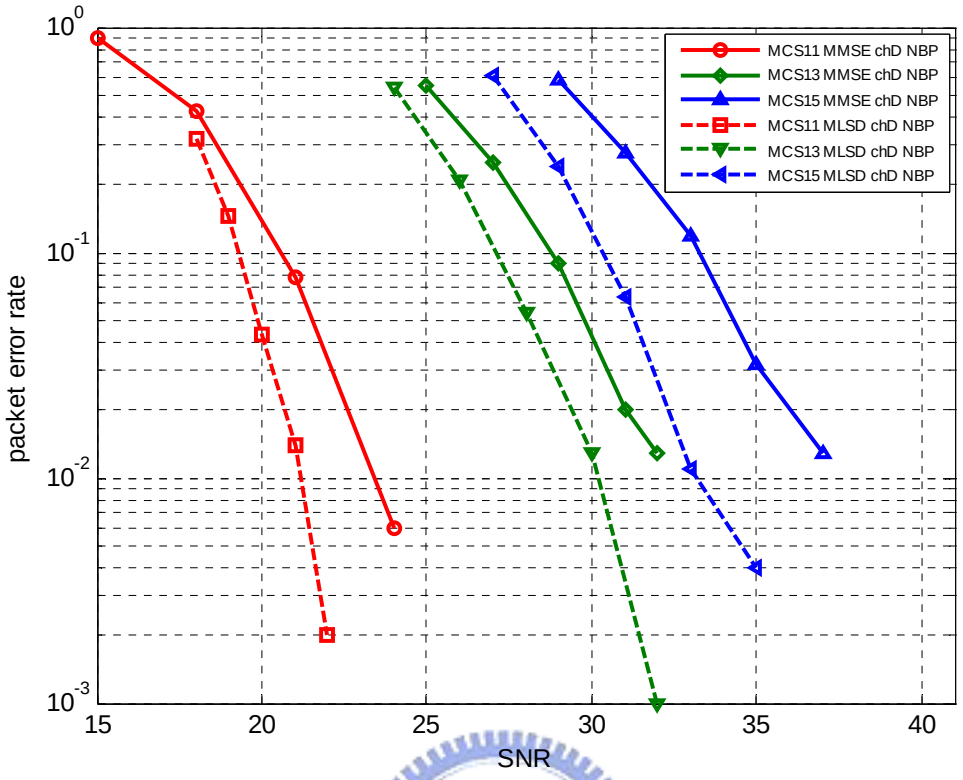


圖 4-24 NBP 解碼在 2x2 通道 D 下模擬結果

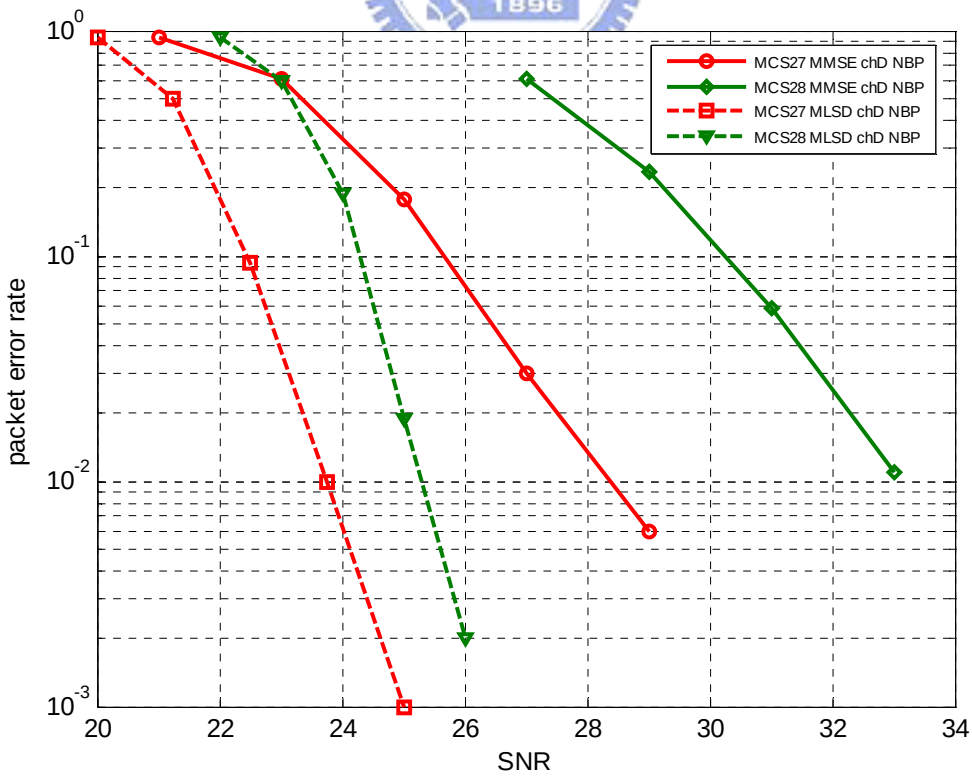


圖 4-25 NBP 解碼在 4x4 通道 D 下模擬結果

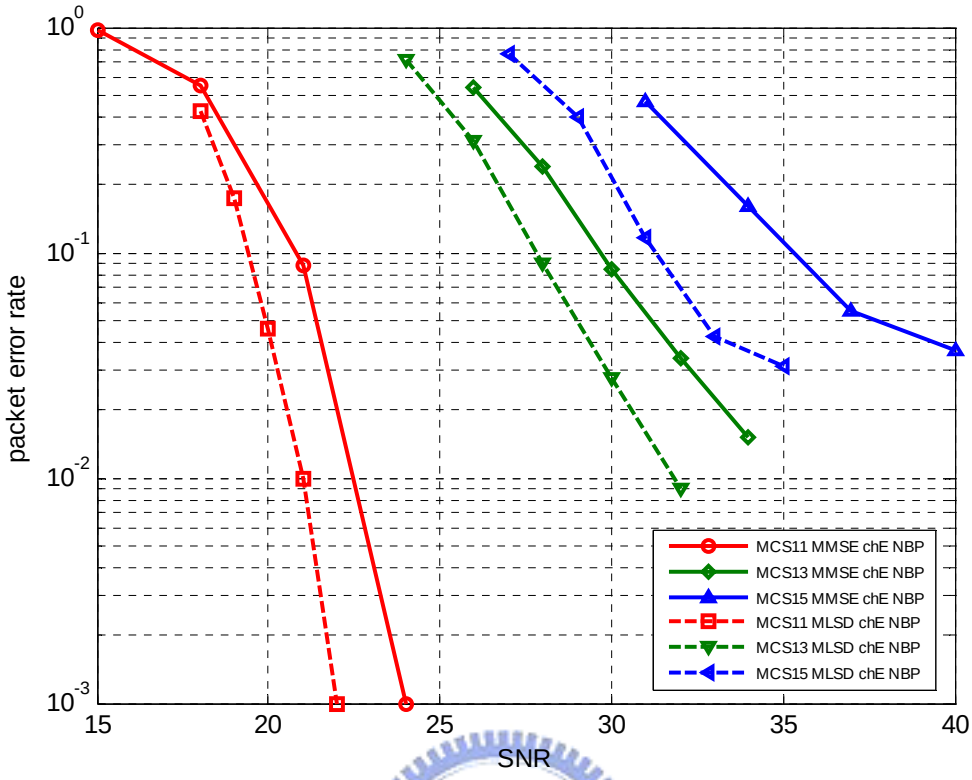


圖 4-26 NBP 解碼在 2x2 通道 E 下模擬結果

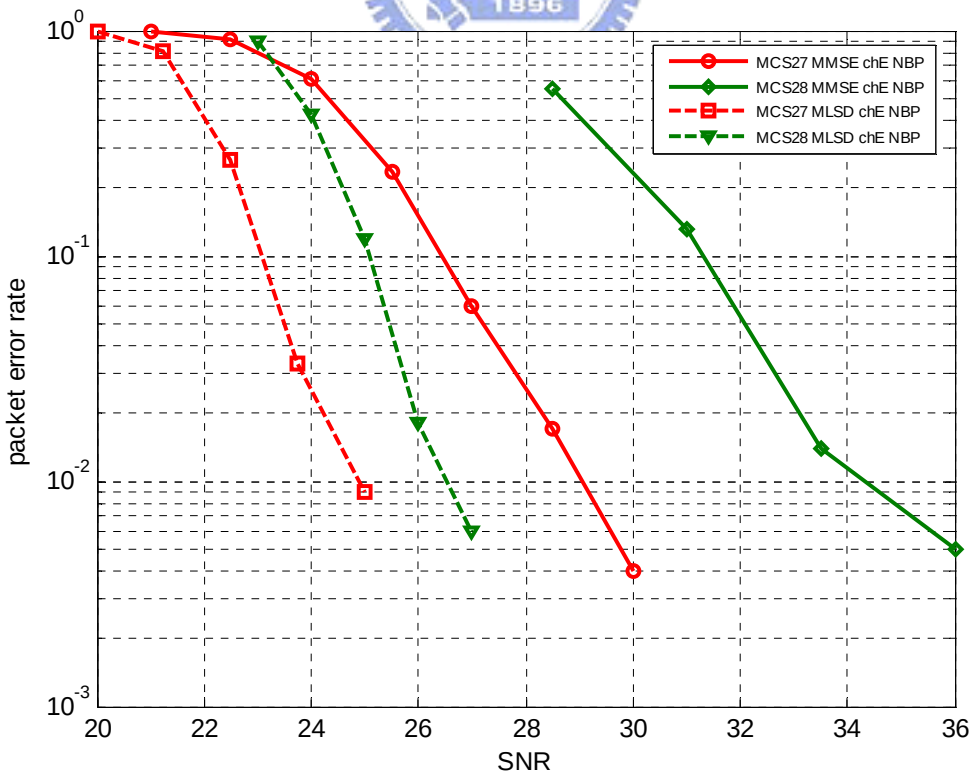


圖 4-27 NBP 解碼在 4x4 通道 E 下模擬結果

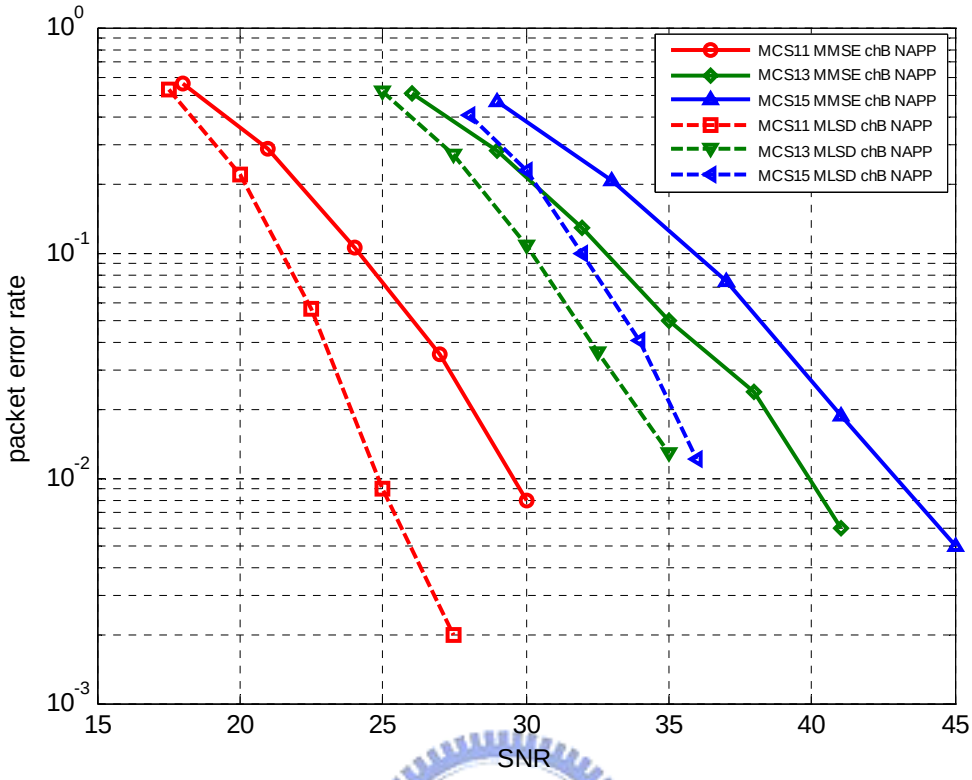


圖 4-28 NAPP 解碼在 2x2 通道 B 下模擬結果

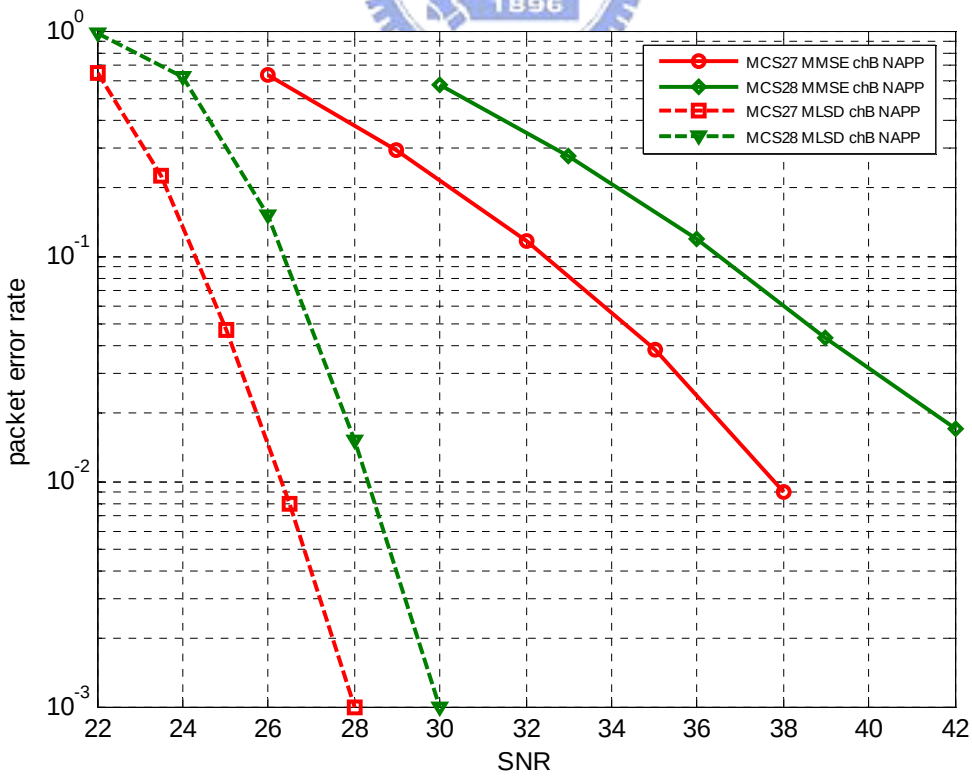


圖 4-29 NAPP 解碼在 4x4 通道 B 下模擬結果

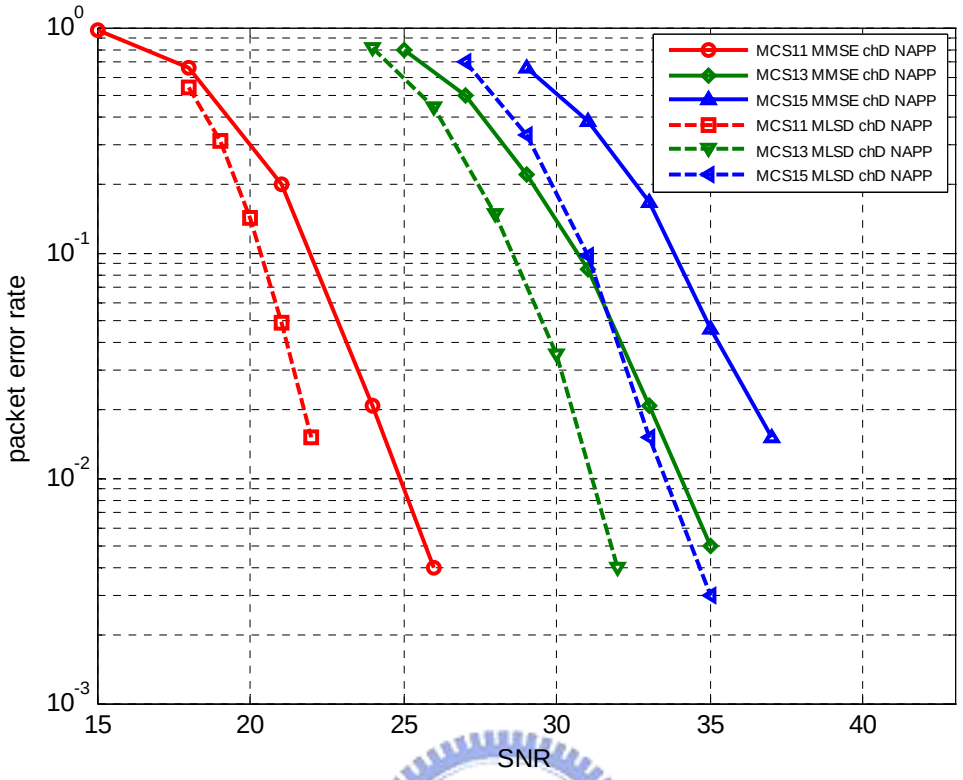


圖 4-30 NAPP 解碼在 2x2 通道 D 下模擬結果

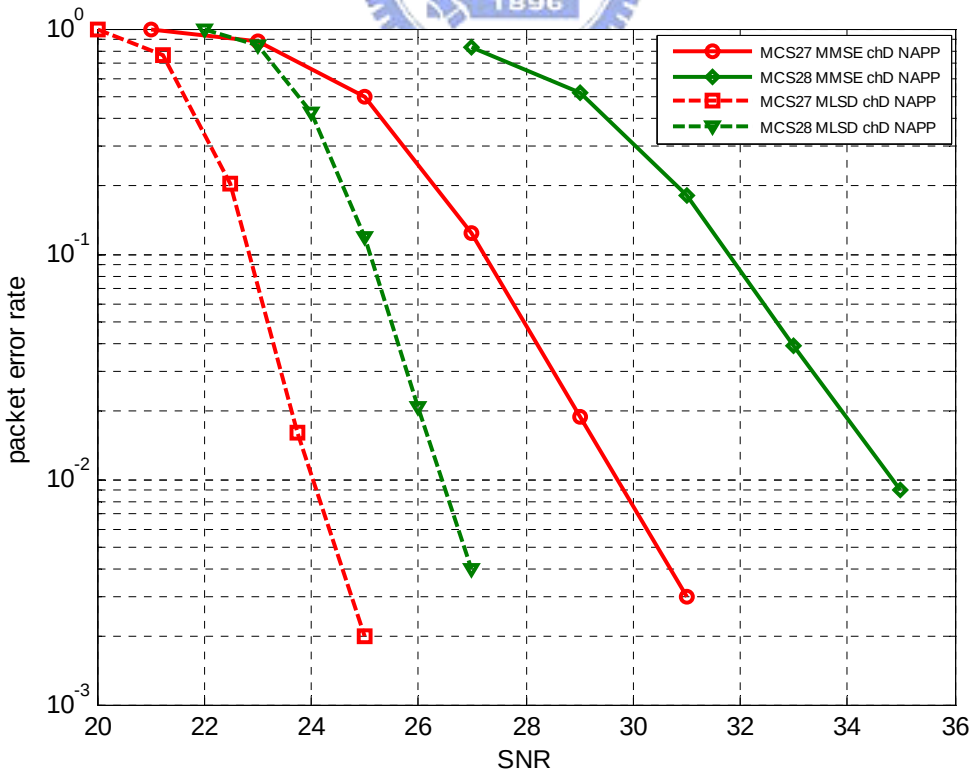


圖 4-31 NAPP 解碼在 4x4 通道 D 下模擬結果

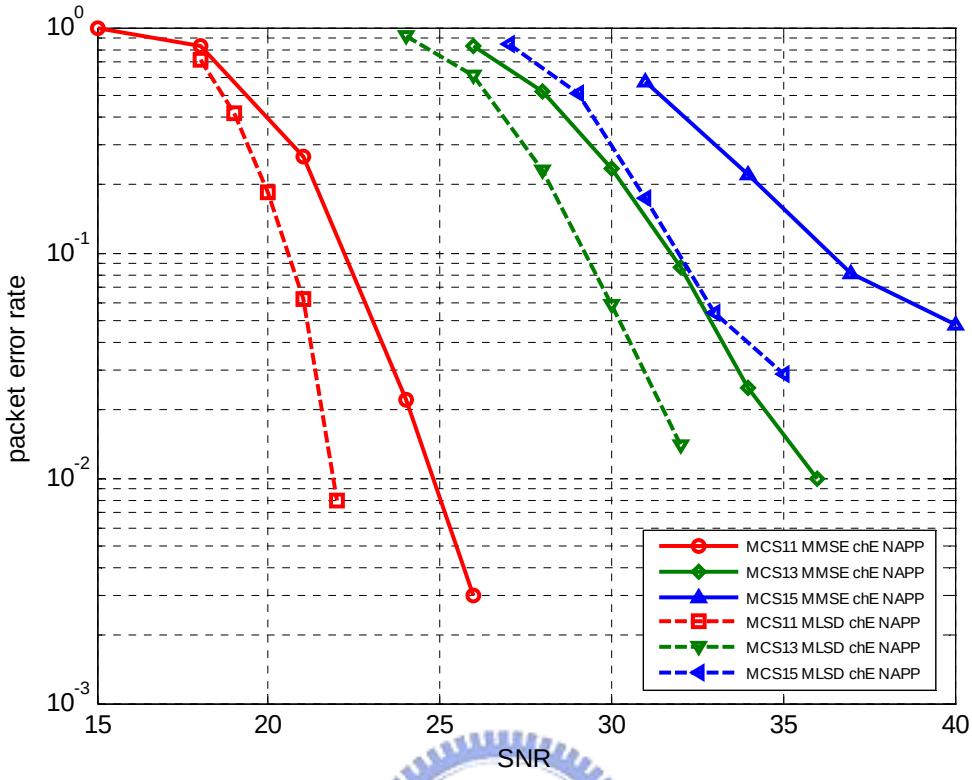


圖 4-32 NAPP 解碼在 2x2 通道 E 下模擬結果

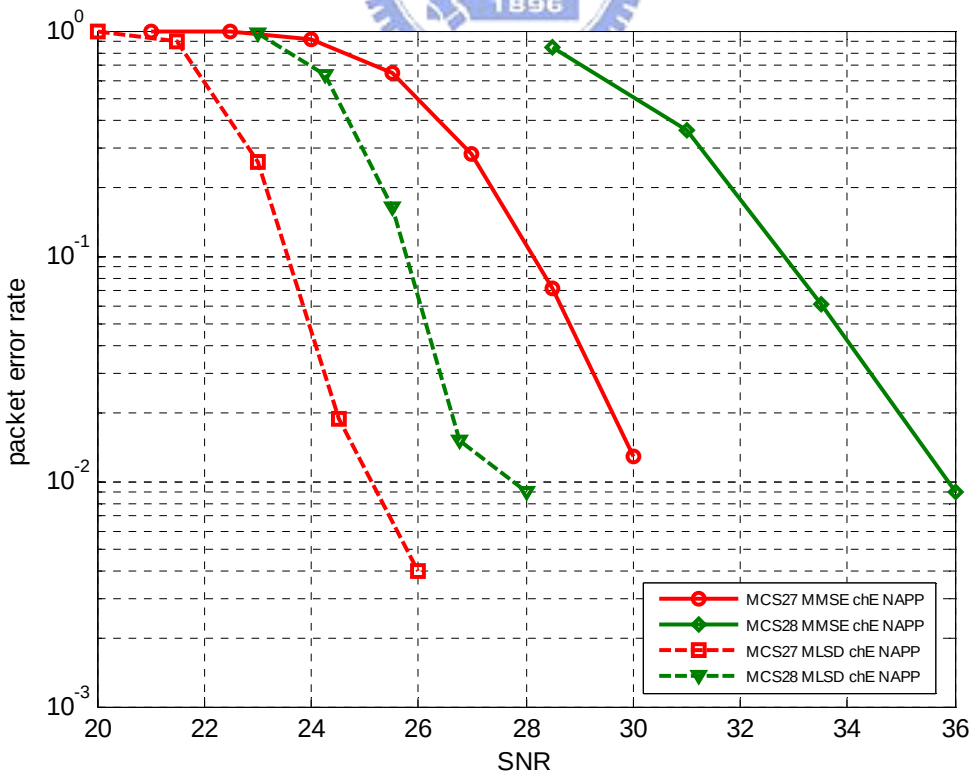


圖 4-33 NAPP 解碼在 4x4 通道 E 下模擬結果

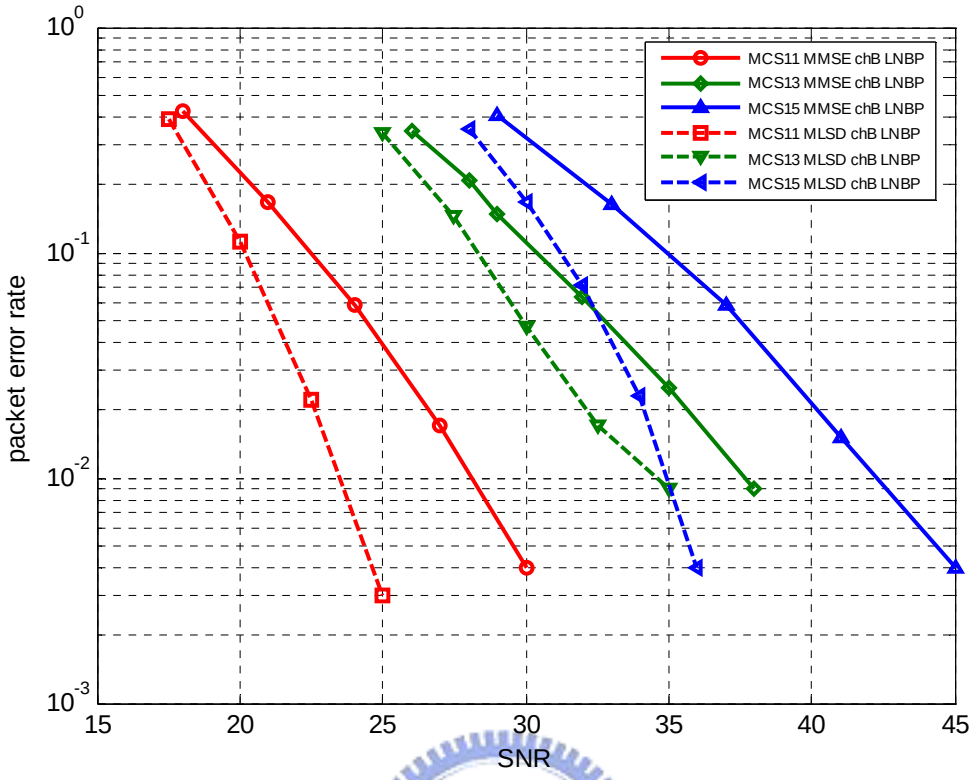


圖 4-34 LNBP 解碼在 2x2 通道 B 下模擬結果

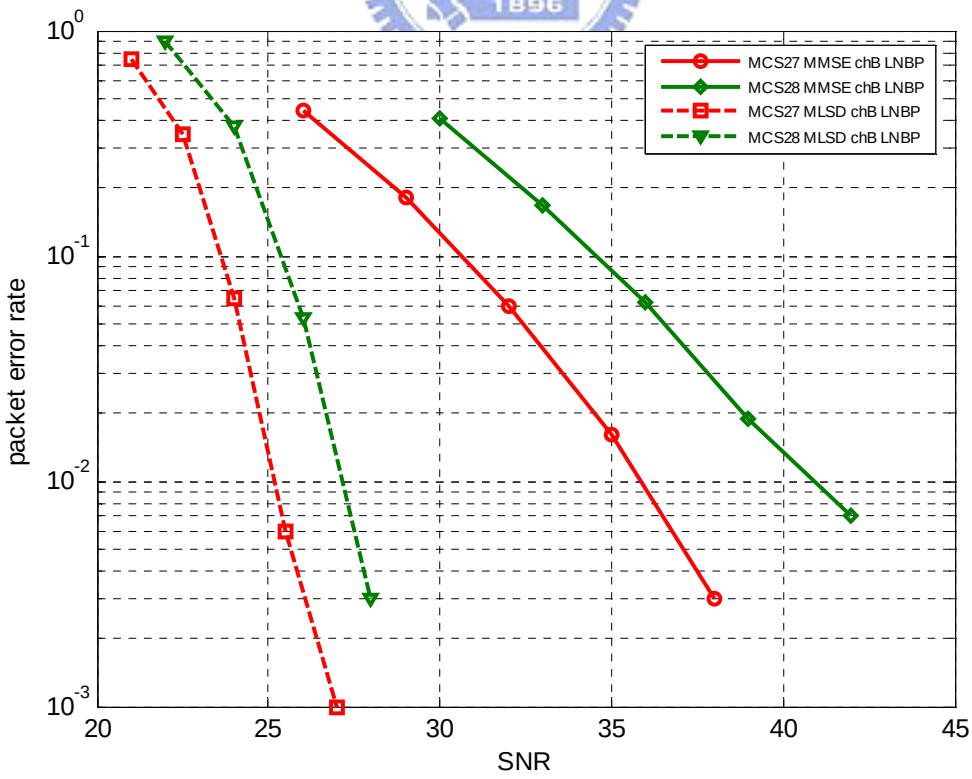


圖 4-35 LNBP 解碼在 4x4 通道 B 下模擬結果

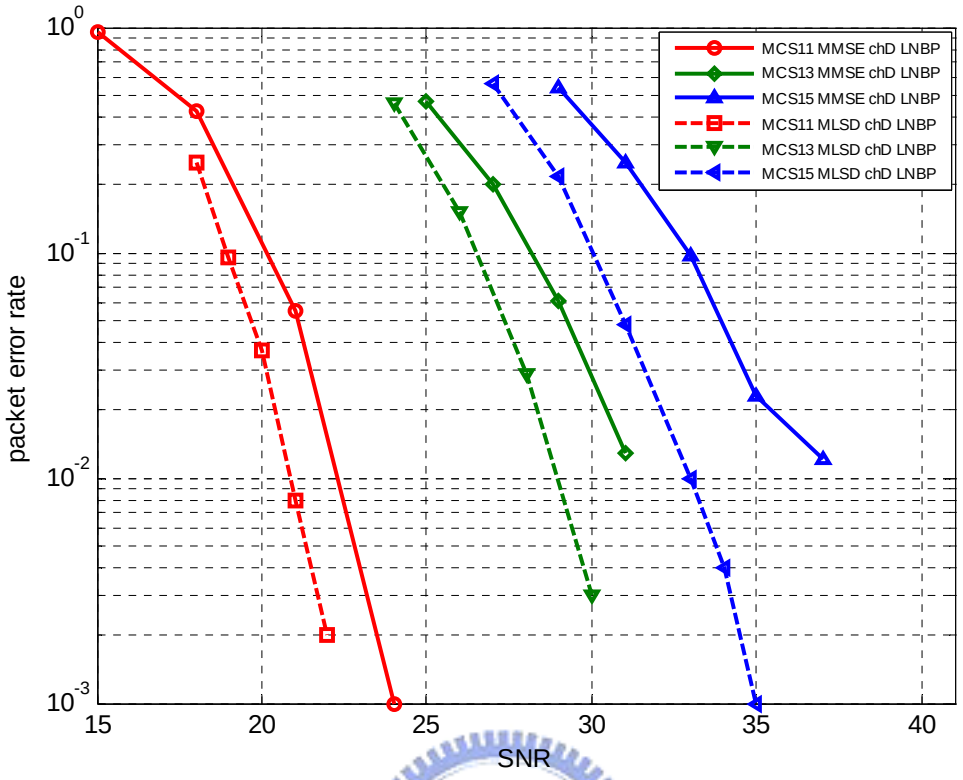


圖 4-36 LNBP 解碼在 2x2 通道 D 下模擬結果

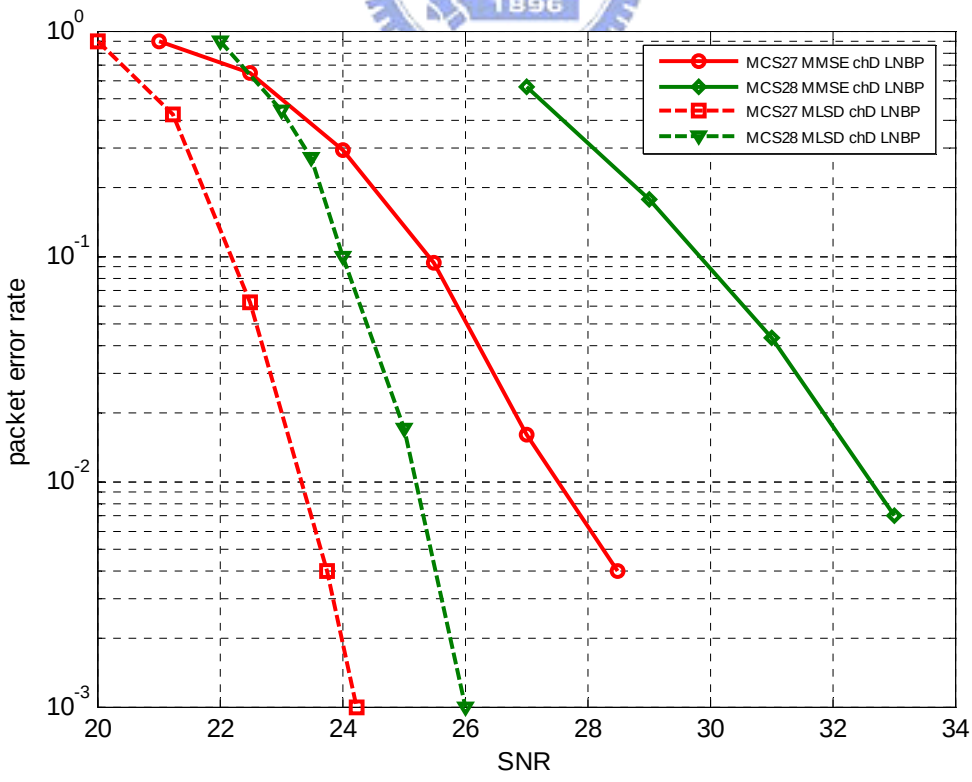


圖 4-37 LNBP 解碼在 4x4 通道 D 下模擬結果

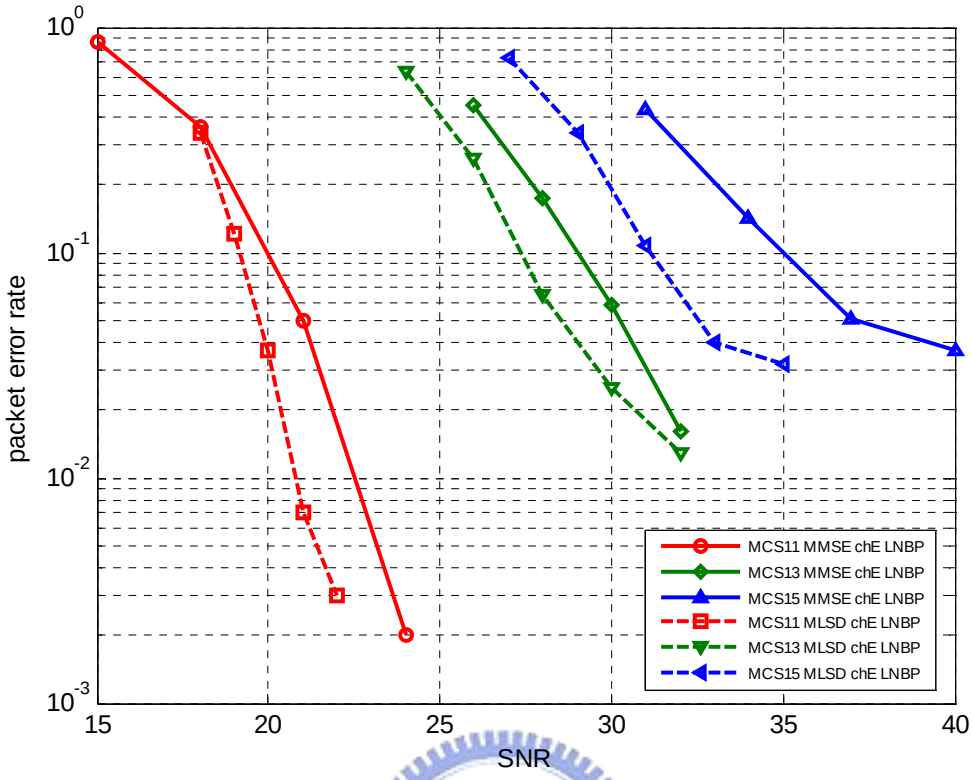


圖 4-38 LNBP 解碼在 2x2 通道 E 下模擬結果

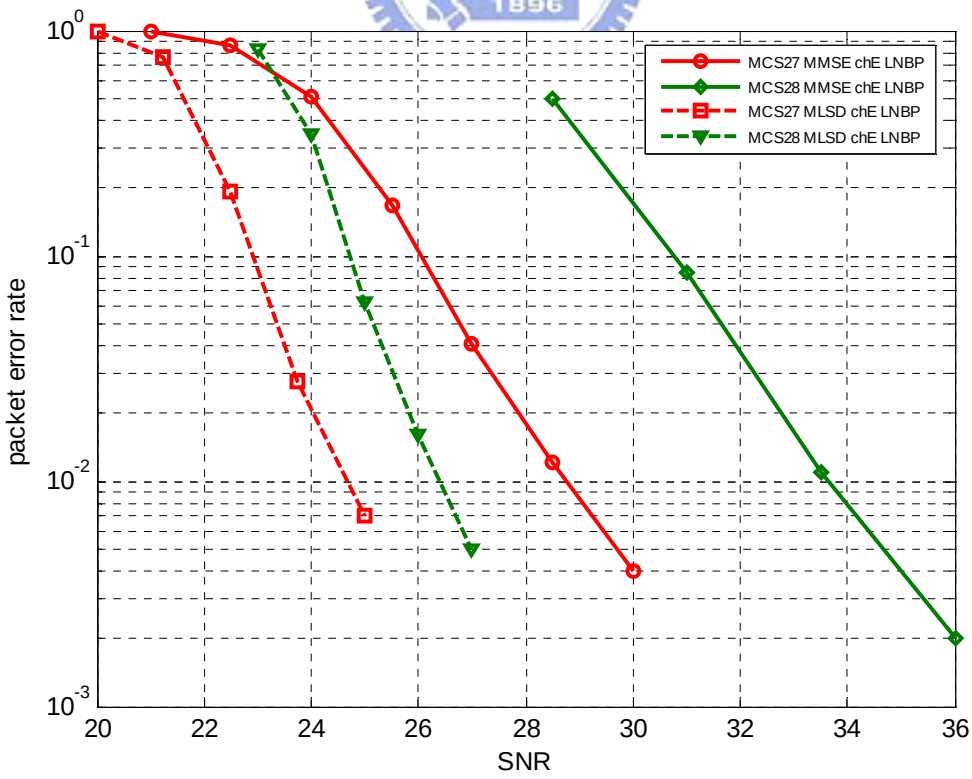


圖 4-39 LNBP 解碼在 4x4 通道 E 下模擬結果

第5章 結論與未來研究目標

5.1 結論

本篇論文主要是在探討 LDPC Codes 解碼器的設計並用於 IEEE 802.11n 及 IEEE 802.16e 之系統中，同時也探討 MIMO 訊號偵測器的設計，特別是在簡化與降低 MIMO Max-Log-MAP 偵測器的運算複雜度的議題上。

在簡化 LDPC Codes 解碼器的設計上，我們使用 Normalized BP-based 演算法與 Normalized APP-based 演算法，並將 Layered BP 演算法和 Normalized BP-based 演算法結合而成 Layered Normalized BP-based 演算法，同時亦藉由置換同位檢查矩陣的列向量，來減輕 Layered BP 演算法對平行處理的破壞。在上述三種演算法中，以 Layered Normalized BP-based 演算法解碼的效能較好，位元錯誤率較低，雖然其分層解碼的架構破壞了其平行處理結構，但其解碼效能的收斂速度卻幾乎是 Normalized BP-based 演算法的兩倍。解碼效能最差的，則是 Normalized APP-based 演算法，但 Normalized APP-based 演算法的運算複雜度卻是三種演算法中最低的。另外，Layered Normalized BP-based 演算法在和 Normalized APP-based 演算法還有一個優點，它們在記憶體的需求上只需 Normalized BP-based 演算法的一半。

在降低 MIMO Max-Log-MAP 偵測器的運算複雜度上，我們使用 List Sphere decoding 演算法，使 s 可能之集合的範圍縮小，讓 Max-Log-MAP 偵測器的運算量得以大幅的降低。此外，亦藉由對搜尋中心 $\hat{s}_\gamma$ 內其元素之絕對值進行排序，再進一步的降低運算量。同時，透過 Max-Log-MAP List Sphere 偵測器，系統的效能亦獲得大幅度的改善，在封包錯誤率同樣為 10^{-1} 的情況下，可分別在 2×2 和 4×4 通道下獲得 3 dB 和 6 dB 以上的改進，在 4×4 B 通道下甚至會有 10 dB 的增益。

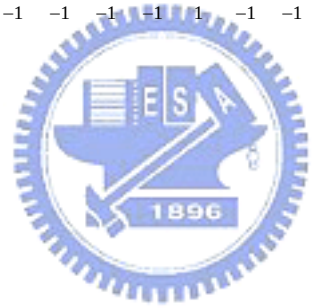
5.2 未來研究目標

MAP 偵測器最大的優勢，便是其可與使用遞迴解碼之解碼器在外部結合成一具有遞迴結構之接收機。我們可藉由 MAP 偵測器將接收訊號轉化為軟性輸出，將之送進遞迴解碼器中解碼，而解碼後所得之軟性輸出可再送進 MAP 偵測器處理，接收訊號之資訊就在 MAP 偵測器與解碼器間重複遞迴，直到求出正確之碼字或達到最大遞迴次數才停止，透過此一架構，可將通道解碼與空時訊號偵測兩者之處理合而為一。然而，為了降低運算複雜度，我們在 MIMO 偵測上只執行一次遞迴偵測，此舉無疑扼殺了此一具有遞迴結構之接收機的優點。另外，List Sphere decoding 演算法仍然有數個問題待解決，其候選列表 $\mathcal{L}$ 內的候選個數與其運算複雜度皆極不固定，動態範圍變化非常大，兩者皆深受通道狀態的影響，這兩個問題皆不利於硬體的設計與實現。在下一階段的研究中，應可將接收機修改為可在偵測器與解碼器之間執行遞迴之系統，同時亦須設法改進候選個數與其運算複雜度不固定之問題，這將是我們未來研究時所要達成之目標。

附錄 A 802.11n 基礎矩陣

Rate1/2 ： $M_b \times N_b = 12 \times 24$

$$\begin{bmatrix}
0 & 0 & -1 & 0 & -1 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
29 & -1 & 0 & 26 & -1 & -1 & 0 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 21 & 0 & -1 & 17 & -1 & -1 & 38 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
43 & -1 & -1 & 30 & -1 & -1 & -1 & 0 & -1 & 41 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
5 & -1 & 1 & -1 & -1 & 20 & 35 & -1 & -1 & 2 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 46 & -1 & -1 & -1 & -1 & 22 & -1 & 40 & 8 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 9 & -1 & -1 & 18 & 13 & -1 & 35 & -1 & 27 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\
2 & -1 & 44 & -1 & -1 & -1 & 27 & -1 & -1 & 25 & 18 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\
33 & 35 & -1 & 29 & -1 & -1 & 16 & -1 & -1 & -1 & -1 & 30 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\
-1 & -1 & -1 & 4 & 4 & -1 & -1 & -1 & 15 & 17 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \\
5 & -1 & -1 & 19 & -1 & 14 & -1 & -1 & -1 & -1 & 11 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\
10 & -1 & -1 & -1 & 21 & -1 & 18 & 8 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$



Rate 2/3 ： $M_b \times N_b = 8 \times 24$

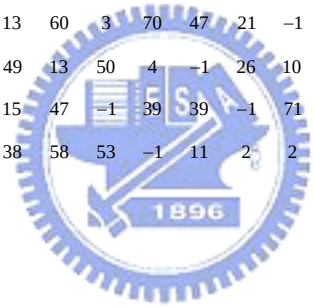
$$\begin{bmatrix}
71 & 40 & -1 & -1 & 11 & 40 & -1 & -1 & -1 & 69 & -1 & 59 & 10 & -1 & -1 & 21 & 46 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
18 & -1 & 35 & 51 & 27 & -1 & -1 & 49 & 17 & -1 & -1 & 37 & 4 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\
42 & 52 & -1 & -1 & 12 & -1 & 54 & -1 & 43 & -1 & 62 & 62 & 52 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\
13 & -1 & 27 & -1 & -1 & -1 & 5 & -1 & 28 & 33 & -1 & -1 & 58 & 47 & 18 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\
-1 & -1 & -1 & 38 & 50 & -1 & 7 & 9 & 34 & -1 & -1 & -1 & 65 & -1 & 17 & 70 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\
71 & -1 & -1 & 39 & 62 & 14 & -1 & -1 & 53 & -1 & 39 & -1 & 2 & 67 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \\
54 & 56 & -1 & -1 & 53 & -1 & -1 & 2 & 66 & 57 & -1 & -1 & -1 & 24 & -1 & 52 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\
17 & -1 & 44 & -1 & 44 & 29 & -1 & -1 & 48 & -1 & 26 & -1 & 39 & -1 & 14 & -1 & 46 & -1 & -1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Rate $3/4$ ： $M_b \times N_b = 6 \times 24$

$$\begin{bmatrix}
0 & 23 & -1 & -1 & 0 & 47 & 53 & 1 & -1 & 5 & -1 & 13 & 66 & 34 & -1 & 62 & 65 & -1 & - & 0 & -1 & -1 & -1 & -1 \\
14 & 57 & 7 & 11 & -1 & -1 & 69 & -1 & 25 & 34 & -1 & 25 & 13 & -1 & 19 & 70 & -1 & 35 & -1 & 0 & 0 & -1 & -1 & -1 \\
59 & 3 & -1 & 18 & -1 & 43 & 48 & 58 & -1 & 29 & -1 & 54 & 49 & -1 & 25 & 0 & 66 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\
15 & 1 & 2 & -1 & -1 & 39 & 19 & -1 & 37 & 64 & 52 & -1 & 28 & 4 & -1 & 21 & 20 & -1 & 0 & -1 & -1 & 0 & 0 & -1 \\
22 & 64 & -1 & 48 & 60 & -1 & 36 & -1 & 15 & 62 & 68 & -1 & 46 & 55 & -1 & 41 & -1 & 7 & -1 & -1 & -1 & -1 & 0 & 0 \\
63 & 36 & 57 & -1 & 55 & -1 & 41 & 34 & -1 & 45 & 46 & -1 & 8 & -1 & 37 & 44 & -1 & 15 & 1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Rate $5/6$ ： $M_b \times N_b = 4 \times 24$

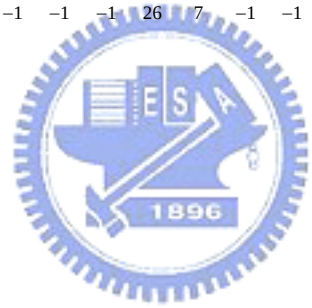
$$\begin{bmatrix}
11 & 26 & 59 & 44 & 36 & 45 & 51 & 32 & 13 & 60 & 3 & 70 & 47 & 21 & -1 & 67 & 52 & 49 & 15 & -1 & 1 & 0 & -1 & -1 \\
39 & 58 & 28 & 60 & 43 & 55 & 30 & 51 & 49 & 13 & 50 & 4 & -1 & 26 & 10 & -1 & 30 & 58 & 28 & 22 & -1 & 0 & 0 & -1 \\
36 & 11 & 35 & 38 & 1 & 5 & 9 & 36 & 15 & 47 & -1 & 39 & 39 & -1 & 71 & 53 & 66 & -1 & 20 & 45 & 0 & -1 & 0 & 0 \\
54 & 67 & 51 & 0 & 15 & 65 & 24 & 17 & 38 & 58 & 53 & -1 & 11 & 2 & 2 & 26 & -1 & 15 & -1 & 6 & 1 & -1 & -1 & 0
\end{bmatrix}$$



附錄 B 802.16e 基礎模型矩陣

Rate1/2 ： $M_{bm} \times N_{bm} = 12 \times 24$

$$\begin{bmatrix}
-1 & 94 & 73 & -1 & -1 & -1 & -1 & -1 & 55 & 83 & -1 & -1 & 7 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 27 & -1 & -1 & -1 & 22 & 79 & 9 & -1 & -1 & -1 & 12 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 24 & 22 & 81 & -1 & 33 & -1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
61 & -1 & 47 & -1 & -1 & -1 & -1 & -1 & 65 & 25 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 39 & -1 & -1 & -1 & 84 & -1 & -1 & 41 & 72 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 46 & 40 & -1 & 82 & -1 & -1 & -1 & 79 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 95 & 53 & -1 & -1 & -1 & -1 & -1 & 14 & 18 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\
-1 & 11 & 73 & -1 & -1 & -1 & 2 & -1 & -1 & 47 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\
12 & -1 & -1 & -1 & 83 & 24 & -1 & 43 & -1 & -1 & -1 & 51 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & 94 & -1 & 59 & -1 & -1 & 70 & 72 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \\
-1 & -1 & 7 & 65 & -1 & -1 & -1 & -1 & 39 & 49 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\
43 & -1 & -1 & -1 & -1 & 66 & -1 & 41 & -1 & -1 & -1 & 26 & 7 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$



Rate 2/3, A code： $M_{bm} \times N_{bm} = 8 \times 24$

$$\begin{bmatrix}
3 & 0 & -1 & -1 & 2 & 0 & -1 & 3 & 7 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 1 & -1 & 36 & -1 & -1 & 34 & 10 & -1 & -1 & 18 & 2 & -1 & 3 & 0 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 12 & 2 & -1 & 15 & -1 & 40 & -1 & 3 & -1 & 15 & -1 & 2 & 13 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\
-1 & -1 & 19 & 24 & -1 & 3 & 0 & -1 & 6 & -1 & 17 & -1 & -1 & -1 & 8 & 39 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\
20 & -1 & 6 & -1 & -1 & 10 & 29 & -1 & -1 & 28 & -1 & 14 & -1 & 38 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\
-1 & -1 & 10 & -1 & 28 & 20 & -1 & -1 & 8 & -1 & 36 & -1 & 9 & -1 & 21 & 45 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \\
35 & 25 & -1 & 37 & -1 & 21 & -1 & -1 & 5 & -1 & -1 & 0 & -1 & 4 & 20 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\
-1 & 6 & 6 & -1 & -1 & -1 & 4 & -1 & 14 & 30 & -1 & 3 & 36 & -1 & 14 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Rate $2/3$, B code： $M_{bm} \times N_{bm} = 8 \times 24$

$$\begin{bmatrix}
2 & -1 & 19 & -1 & 47 & -1 & 48 & -1 & 36 & -1 & 82 & -1 & 47 & -1 & 15 & -1 & 95 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 69 & -1 & 88 & -1 & 33 & -1 & 3 & -1 & 16 & -1 & 37 & -1 & 40 & -1 & 48 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\
10 & -1 & 86 & -1 & 62 & -1 & 28 & -1 & 85 & -1 & 16 & -1 & 34 & -1 & 73 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\
-1 & 28 & -1 & 32 & -1 & 81 & -1 & 27 & -1 & 88 & -1 & 5 & -1 & 56 & -1 & 37 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\
23 & -1 & 29 & -1 & 15 & -1 & 30 & -1 & 66 & -1 & 24 & -1 & 50 & -1 & 62 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\
-1 & 30 & -1 & 65 & -1 & 54 & -1 & 14 & -1 & 0 & -1 & 30 & -1 & 74 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \\
32 & -1 & 0 & -1 & 15 & -1 & 56 & -1 & 85 & -1 & 5 & -1 & 6 & -1 & 52 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\
-1 & 0 & -1 & 47 & -1 & 13 & -1 & 61 & -1 & 84 & -1 & 55 & -1 & 78 & -1 & 41 & 95 & -1 & -1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Rate $3/4$, A code： $M_{bm} \times N_{bm} = 6 \times 24$

$$\begin{bmatrix}
6 & 38 & 3 & 93 & -1 & -1 & -1 & 30 & 70 & -1 & 86 & -1 & 37 & 38 & 4 & 11 & -1 & 46 & 48 & 0 & -1 & -1 & -1 & -1 \\
62 & 94 & 19 & 84 & -1 & 92 & 78 & -1 & 15 & -1 & -1 & 92 & -1 & 45 & 24 & 32 & 30 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\
71 & -1 & 55 & -1 & 12 & 66 & 45 & 79 & -1 & 78 & -1 & -1 & 10 & -1 & 22 & 55 & 70 & 82 & -1 & -1 & 0 & 0 & -1 & -1 \\
38 & 61 & -1 & 66 & 9 & 73 & 47 & 64 & -1 & 39 & 61 & 43 & -1 & -1 & -1 & -1 & 95 & 32 & 0 & -1 & -1 & 0 & 0 & -1 \\
-1 & -1 & -1 & -1 & 32 & 52 & 55 & 80 & 95 & 22 & 6 & 51 & 24 & 90 & 44 & 20 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\
-1 & 63 & 31 & 88 & 20 & -1 & -1 & -1 & 6 & 40 & 56 & 16 & 71 & 53 & -1 & -1 & 27 & 26 & 48 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Rate $3/4$, B code： $M_{bm} \times N_{bm} = 6 \times 24$

$$\begin{bmatrix}
-1 & 81 & -1 & 28 & -1 & -1 & 14 & 25 & 17 & -1 & -1 & 85 & 29 & 52 & 78 & 95 & 22 & 92 & 0 & 0 & -1 & -1 & -1 & -1 \\
42 & -1 & 14 & 68 & 32 & -1 & -1 & -1 & -1 & 70 & 43 & 11 & 36 & 40 & 33 & 57 & 38 & 24 & -1 & 0 & 0 & -1 & -1 & -1 \\
-1 & -1 & 20 & -1 & -1 & 63 & 39 & -1 & 70 & 67 & -1 & 38 & 4 & 72 & 47 & 29 & 60 & 5 & 80 & -1 & 0 & 0 & -1 & -1 \\
64 & 2 & -1 & -1 & 63 & -1 & -1 & 3 & 51 & -1 & 81 & 15 & 94 & 9 & 85 & 36 & 14 & 19 & -1 & -1 & -1 & 0 & 0 & -1 \\
-1 & 53 & 60 & 80 & -1 & 26 & 75 & -1 & -1 & -1 & -1 & 86 & 77 & 1 & 3 & 72 & 60 & 25 & -1 & -1 & -1 & -1 & 0 & 0 \\
77 & -1 & -1 & -1 & 15 & 28 & -1 & 35 & -1 & 72 & 30 & 68 & 85 & 84 & 26 & 64 & 11 & 89 & 0 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Rate $5/6$ ： $M_{bm} \times N_{bm} = 4 \times 24$

$$\begin{bmatrix}
125 & 55 & -1 & 4 & 7 & 4 & -1 & 91 & 84 & 8 & 86 & 52 & 82 & 33 & 5 & 0 & 36 & 20 & 4 & 77 & 80 & 0 & -1 & -1 \\
-1 & 6 & -1 & 36 & 40 & 47 & 12 & 79 & 47 & -1 & 41 & 21 & 12 & 71 & 14 & 72 & 0 & 44 & 29 & 0 & 0 & 0 & 0 & -1 \\
51 & 81 & 83 & 4 & 67 & -1 & 21 & -1 & 31 & 24 & 91 & 61 & 81 & 9 & 86 & 78 & 60 & 88 & 67 & 15 & -1 & -1 & 0 & 0 \\
68 & -1 & 50 & 15 & -1 & 36 & 13 & 10 & 11 & 20 & 53 & 90 & 29 & 92 & 57 & 30 & 84 & 92 & 11 & 66 & 80 & -1 & -1 & 0
\end{bmatrix}$$

參考文獻

[1]. C. E. Shannon, “A Mathematical Theory of Communication,” *Bell Syst. Tech. J.*, pp. 379-423 (Part 1); pp. 623-56 (Part 2), July 1948.

[2]. E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York, 1968; Rev. ed., Aegean Park Press, Laguna Hills, N.Y., 1984.

[3]. R. W. Hamming, “Error Detecting and Error Correcting Codes,” *Bell Syst. Tech. J.*, 29: 147-60, April 1950.

[4]. P. Elias, “Coding for Noisy Channels,” *IRE Conv. Rec., p. 4*: 37-47, 1955.

[5]. I. S. Read and G. Solomon, “Polynomial Codes over Certain Fields,” *J. Soc. Ind. Appl. Math.*, 8: 300-304, June 1960.

[6]. C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes,” *Proc. IEEE Intl. Conf. Commun. (ICC93)*, pp. 1064-70, Geneva, Switzerland, May 1993.

[7]. R. G.. Gallager, “Low Density Parity Check Codes,” IRE Trans. Inform. Theory, IT-8: 21-28, January 1962.

[8]. D. J. C. Mackay and R. M. Neal, “Near Shannon Limit Performance of Low Density Parity Check Codes,” *Electron. Lett.*, 32 (18): 1645-46, 1996.

[9]. D. J. C. Mackay, “Good Error-Correcting Codes Based on Very Sparse Matrices,” *IEEE Trans. Inform. Theory*, 45 (2): 399-432, March 1999.

[10]. J. Chen and M. P. C. Fossorier, “Near Optimum Universal Belief Propagation Based Decoding of Low-Density Parity Check Codes,” *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 406-414, March 2002.

[11]. J. Chen and M. P. C. Fossorier, “Decoding Low-Density Parity Check Codes With Normalized APP-Based Algorithm,” in *Proc. IEEE Globecom*, San Antonio,

TX, Nov. 2001, pp. 102-104, Nonember 2001.

[12]. D. E. Hocevar, DSP. Solutions R&D Center, Texas Instruments, Dallas, TX, USA. “A reduced Complexity Decoder Architecture via Layered Decoding of LDPC Codes,” *IEEE SIPS*, pp. 107-112, December 2004.

[13]. J. Benesty, Y. Huang, and J. Chen, “A Fast Recursive Algorithm for Optimum Sequential Signal Detection in a BLAST System,” *IEEE transactions on Signal Processing*, vol. 51, no. 7, July 2003.

[14]. F. Tosato and P. Bisaglia, “Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2,” *Proc. IEEE Int’l. Conf. on Comm. 2002*, vol. 2, pp. 664-668, 2002.

[15]. “TGn Sync proposal technical specification, ” *TGn Sync*, Mar. 2005.

[16]. *IEEE 802.11a Stand., ISO/IEC 8802-11:1999/Amd 1:2000(E)*

[17]. *IEEE 802.11n Wireless LAN TGN Channel model*, May 2004.

[18]. *IEEE Std 802.16e and IEEE Std 802.16-2004/Cor 1-2005*, “Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems,” February 2006.

[19]. *IEEE Std 802.16-2004*, “Part 16: Air Interface for Fixed Broadband Wireless Access Systems,” October 2004.

[20]. J. Choi, “On the Partial MAP Detection with Applications to MIMO Channels,” *IEEE Trans. Signal Process.*, vol. 53, pp. 158–167, January 2005.

[21]. H. Vikalo and B. Hassibi, “Maximum Likelihood Sequence Detection of Multiple Antenna Systems over Dispersive Channel via Sphere Decoding,” *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 5, pp. 523-531, 2002.

[22]. B. Hochwald and S. ten Brink, “Achieving Near-Capacity on a Multiple-Antenna Channel,” *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389-399, March 2003.

[23]. H. Vikalo, B. Hassibi, and T. Kailath, “Iterative Decoding for MIMO Channels

Via Modified Sphere Decoding," *IEEE Trans. on Wireless Commun.*, vol. 3, no. 6, pp. 2299-2311, November 2004.

[24]. Mong Suan Yee, "Max-Log-MAP Sphere Decoder," IEEE ICASSP 2005, pp. III-1013-III-1016, 2005.

[25]. O. Damen, A. Chkeif, and J. C. Belfiore, "Lattice Code Decoder for Space-Time Codes," *IEEE Commun. Letter*, vol. 4, no. 5, pp. 161-163, May 2000.

簡歷

姓　　名：　黃冠榮

性　　別：　男

出生日期：　民國 69 年 11 月 27 日

出生地　：　台北縣

學　歷　：

台北市立光復國小　(1987.9~1993.6)

台北市立介壽國中　(1993.9~1996.6)

國立台灣師大附中　(1996.9~1999.6)

國立暨南國際大學電機工程學系　(1999.9~2003.6)

國立交通大學電信工程研究所碩士班　(2004.9~2006.7)

公元 2006 年 7 月獲得碩士學位