國　立　交　通　大　學

電信工程學系碩士班

碩士論文

馬可夫鏈-蒙地卡羅方法應用於多輸入多輸出系統進行資料偵測之研究

A Study on applying Markov Chain Monte Carlo Method for Data Detection in MIMO systems

研究生：余建勳

指導教授：黃家齊　　博士

中華民國　九十七年七月

馬可夫鏈-蒙地卡羅方法應用於多輸入多輸出系統進行資料偵測之研究

A Study on applying Markov Chain Monte Carlo Method for Data Detection in MIMO systems

研 究 生：余建勳　　Student：Chien-Hsun Yu

指導教授：黃家齊　　Advisor：Dr. Chia-Chi Huang

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 and Computer Engineering

National Chiao Tung University

In Partial Fulfillment of the Requirements

for the Degree of

Mater of Science

in Communication Engineering

July 2007

Hsinchu, Taiwan , Republic of China

中華民國九十七年七月

馬可夫鏈-蒙地卡羅方法應用於多輸入多輸出系統進行資料偵測之研究

學生：余建勳　　指導教授：黃家齊

國立交通大學電工程學系　碩士班

摘　　要

在現代的通訊系統中，由於所需要的傳輸速率越來高，因此有許多高階的調變方法，例如 **16QA**M，64QAM 等方法都被應用到現有以 MIMO 爲基本架構的通信系統上，如此可以再不改變系統架構的情況下提升資料傳輸的速度。然而當調變越來越複雜，原本的接收方法便會顯得過於複雜，因此我們需要找尋新的接收方式來進行資料的接收。馬可夫鏈-蒙地卡羅方法利用簡單的統計學原理，來進行資料的解調，以達到降低複雜度的目的。同時再搭配一個簡單的軟式解碼器，利用遞迴地傳遞軟資訊，可以更改善其錯誤率的表現。

A Study on applying Markov Chain Monte Carlo Method for Data Detection in MIMO systems

Student：Chien-Hsun Yu　　Advisor：Dr. Chia-Chi Huang

Department of Communication Engineering

National Chiao Tung University

ABSTRACT

In modern communication systems, in order to achieve higher bit rate, more complex modulation methods such as 16QAM, 64QAM, are applied to an MIMO based wireless communication system. As a result, the complexity of the receiver becomes higher than usual. Markov Chain Monte Carlo methods are very simple and thus provides a very efficient scheme for data detection . The complexity of the algorithm grows linearly either with the number of the antenna or the complexity of the modulation scheme. If we apply the soft-in soft-out decoder after the MCMC detector, the performance can be comparable with other good yet more complex receiver schemes.

誌謝

感謝黃家齊教授兩年的指導，並在碩士論文上提供許多的建議與觀念，使這一篇論文的完成更加的順利與完整。此外，更感謝老師介紹耶穌給我認識，讓我在低潮的時候有力量，失意的時候不沮喪，我知道這些考驗是神賜給我的磨難，他希望我更堅強。

這一篇論文的完成要特別感謝古孟霖，黃朝旺，邱麟凱，以及鄭有財等四位學長。學長們在這兩年來不辭辛苦的解答我的問題與疑惑，並且提供許多論文的方向與提點我應該注意的細節，更重要的是讓我了解此研究在通訊領域上相關的應用與發展方向。

感謝這兩年一同患難的實驗室同學陸裕威、陳文娟、李思潔，沒有他們幫忙分享生活及研究上的壓力，我也沒有辦法調適好自己的身心，可以每天面對研究的挑戰。

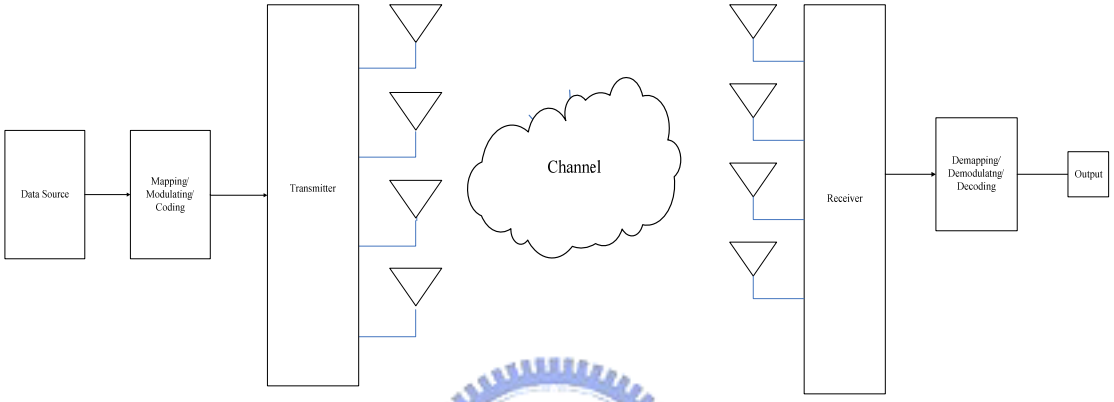
最後，感謝我的父母及家人，在我心灰意冷時，總是給我最堅強的支持。

圖目錄

表目錄

第一章　多輸入多輸出系統簡介

爲了追求更高的資料傳輸速率，現代的通訊系統多採用了多輸入多輸出(Multi input Multi Output)的系統架構，即在傳送及接收端都有複數根天線來執行傳送與接收的工作。下圖即爲一多輸入多輸出之系統架構示意圖。



. 圖 1　多輸入多輸出系統架構圖

接下來我們將探討多輸入多輸出系統架構的系統模型(System Model)。一般而言，我們假設傳送與接收間的關係如下式

$$\mathbf{Y}_{N_r\times 1} = \mathbf{H}_{N_r\times N_t}\mathbf{X}_{N_t\times 1} + \mathbf{n}_{N_r\times 1}$$

其中 N_t 爲傳送端天線的數目， N_r 爲接收端天線的數目。 $\mathbf{X}$ 爲我們所送的資料矩陣， $\mathbf{Y}$ 則爲經過通道所接收到的訊息。 $\mathbf{H}$ 代表通道矩陣，其中的元素 $h_{i,j}$ 代表由傳送端的第 i 根天線到接收端的第 j 根天線所經過的通道，通常是一個複數高斯隨機變數。

使用多根天線來進行傳輸有什麼好處呢？首先得到的好處就是空間上的多工(Spatial Multiplexing)，以及多樣性(Diversity)。因每個通道都是彼此獨立的，因此相同的信號會經過不同衰減(Fading)的通道而抵達接收端，只要在接收端所收到的訊號當中有一個是經過訊雜比(Signal to Noise Ratio)較佳的通道，則我們可以順利地得到我們當初所傳送的信號。此外，由消息理論我們可以知道，一個通道的通道容量(Channel Capacity)是由下式決定

$$C = E_H\{\log_2[\det(\mathbf{I} + \frac{\rho}{n_R}\mathbf{H}\mathbf{H}^\dagger)]\} \propto \min\{n_R, n_T\}$$

其中，ρ 代表通道的訊雜比，n_R 指的是接收端的天線個數。且由上式可以看出通道容量是正比於傳送與接收的天線個數的。

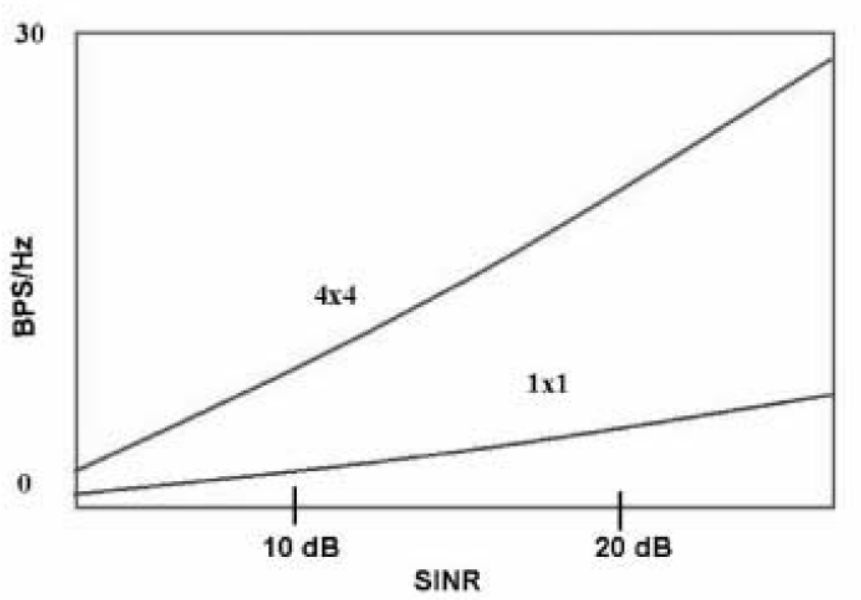


圖 2　1 傳 1 收與 4 傳 4 收通道容量比較圖 [摘自 reference 16]

一般而言，常見於多輸入多輸出系統解調的有如下幾種方法：ML、ZF、VBLAST+OSIC 等，以下就一一的來做介紹。最大概似法(Maximum Likelihood, ML)是一種暴力式的搜尋法，此方法須將所有可能的解代入判別式中，並一一計算其概似函數(Likelihood function)的大小，而概似函數其值最大者即爲最有可能的解。以前面的系統架構爲例，最有可能的解 $\mathbf{x}$ 爲符合判別式

$$\arg\min_{\mathbf{x}\in\Lambda} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|$$

其中 Λ 爲所有可能解所形成之集合。Zero Forcing(ZF)方法則是簡單地利用矩陣的數學特性將通道矩陣之假反矩陣(Pseudo Inverse)乘上所收到的信號 $\mathbf{y}$ ，以去除信號間彼此的干擾，即

$$\hat{\mathbf{x}} = (\mathbf{H}^\dagger\mathbf{H})^{-1}\mathbf{H}^\dagger\mathbf{y}$$

其中 $\hat{\mathbf{x}}$ 爲所預估的信號，†則代表對一矩陣做共軛加轉置的運算。ZF 的缺點在於

其會將所收到的雜訊放大，造成錯誤率的增加。而 VBLAST+OSIC 則是先找出通道訊雜比最佳的那一個訊號來求解，完成後再將其由所收到的信號中扣除，以降低不同資料信號間的影響。以前面的例子爲例，我們以通道矩陣 **H** 中行向量的絕對值最大者當作第一個解調的資料信號，因其通道的訊雜比最佳，錯誤機率相對較低。其缺點在於若前面的信號出現錯誤，則連帶地會影響後面信號的解調，造成一連串的錯誤(Burst Error)。

當我們在解碼的同時，若是能夠針對每個位元求其軟式資訊(Soft Information)，並且重複利用這些資訊以增進解碼效率的解碼器，稱爲渦輪式解碼器(Turbo Decoder)，其架構如下圖所示

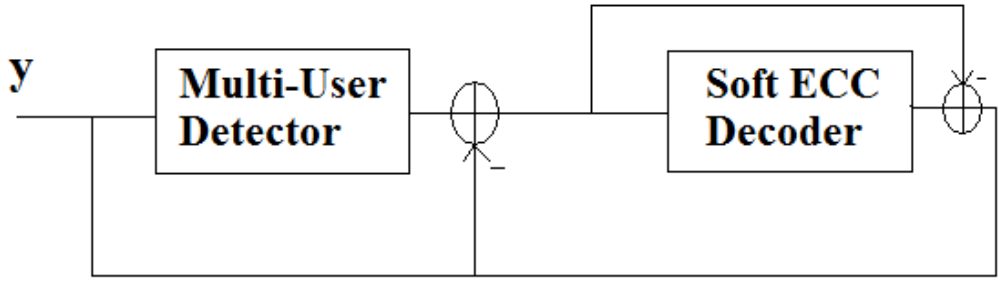


圖 3 一個渦輪式解碼系統示意圖

前面一級我們稱爲多使用者偵測(Multi-User Detector)，其目的在於分別求出每一個位元的軟式資訊。此資訊可以提供較接收訊號 **y** 更爲可靠的資訊給後面一級的軟式錯誤更正碼解碼器，並利用錯誤更正碼的能力將前一級尚未更正的錯誤給更正回來。而錯誤更正碼所輸出的軟式資訊也可以視爲第一級每個位元的事前機率。這也是爲何渦輪式解碼器可以改善錯誤率表現的原因。

在本篇論文中，在多使用者偵測這一級我們使用兩種方法，分別爲馬可夫鏈-蒙地卡羅法以及表列式球狀解碼法，並分別在第二章及第四章說明之；在錯誤更正碼這個部分使用的是軟式維特比演算法(Soft Output Viterbi Algorithm)，我們將在第三章中做說明；第五章則針對渦輪式解碼器的運作模式做一個詳細的解說，包括在前後兩級間，軟式資訊是如何互相傳送。第六章則是模擬的結果，可以觀察在同樣的錯誤率表現下，馬可夫鏈-蒙地卡羅方法的複雜度可以降低多少。

第二章　馬可夫鏈-蒙地卡羅方法

貝氏估計(Bayesian approach)是一種常使用的資料偵測方法，其主要限制在於如何求出後驗機率(posterior probability)，通常後驗機率都需要經過高維度的積分才能求出，而這樣的計算通常對於現有的硬體而言過於複雜，也因此有許多計算高維度積分的方法被提了出來，在此我們專注於探討馬可夫鏈-蒙地卡羅(Markov Chain Monte Carlo, MCMC) 方法。其主要概念爲針對某個複雜的機率分佈 $f(x)$ 來做取樣，藉由建立一個馬可夫鏈，其平衡機率分佈(Equilibrium Distribution)亦爲 $f(x)$ 。MCMC 方法中的其中一種演算法爲 Metropolis-Hasting 演算法，此法在一開始的時候被物理學家用於計算複雜的積分，像是一些熱力學的問題。在 1990, Gelfand 與 Smith 提出了一個可以很容易的實踐 MCMC 的方法[4]，稱爲 Gibbs sampler, 並且在之後被廣泛的運用在各種貝氏機率的問題上。

2.1 蒙地卡羅積分

最原始的蒙地卡羅方法是用來計算複雜的積分，舉例來說，我們希望計算一個積分如下

$$\int_a^b h(x)dx \tag{2.1}$$

如果我們可以將 $h(x)$ 分解成一個函數 $f(x)$ 與一個機率密度函數(probability density function) $p(x)$ ， $p(x)$ 被定義在區間[a,b]中，則由式(2.1)可推得

$$\int_a^b h(x)dx = \int_a^b f(x)p(x)dx = E_{p(x)}[f(x)] \tag{2.2}$$

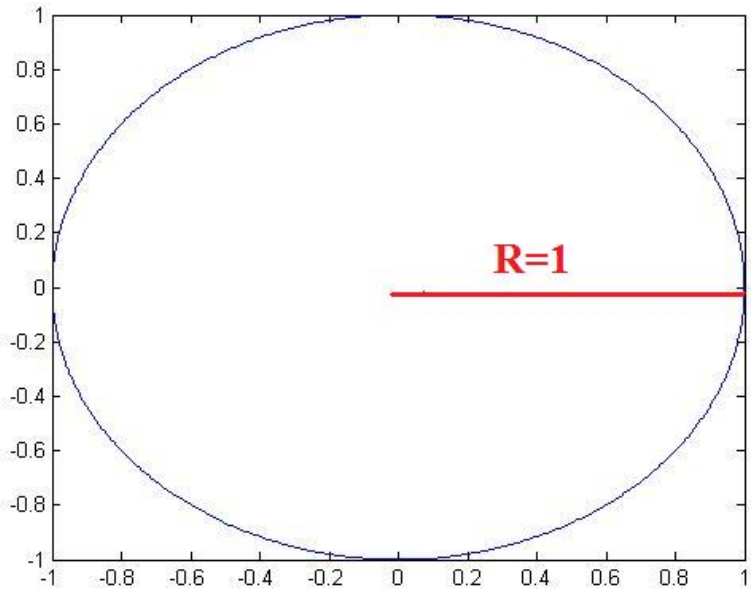
由上式可知積分可以被表示成對 $f(x)$ 取期望值，而每一個變數 x 的機率密度函數都是 $p(x)$ 。換句話說，如果我們用 $p(x)$ 產生一組取樣點，由蒙地卡羅模擬可以得到

$$\int_a^b h(x)dx = E_{p(x)}[f(x)] \cong \frac{1}{n}\sum_{i=1}^{n} f(x_i) \qquad (2.3)$$

式(2.3)亦被稱爲蒙地卡羅積分(Monte Carlo Integration)。

蒙地卡羅積分可以被用來近似貝氏分析所需要的後驗機率或是邊際機率(marginal posterior)分佈。考慮一個積分 $I(y) = \int f(y \mid x)\, p(x)dx$ ，則我們可以近似其爲 $\hat{I}(y) = \frac{1}{n}\sum_{i=1}^{n} f(y \mid x_i)$ ， x_i 爲 $p(x)$ 的取樣點。另外一個由取樣點來求原始函數值的例子是我們可以利用取樣來求出圓周率 π 值，過程如下。

Ex: 如下圖爲一個半徑爲1的圓，正方形的部分爲我們的取樣空間



接下來我們任意產生取樣點，並且記錄取樣點落在圓內與落在圓外到方型間的點數比，可以得到如下的結果:

產生10個取樣點，落在圓內的有6個 $\Rightarrow \frac{\pi}{4} = 0.6 \quad \pi = 2.4$

產生100個取樣點，落在圓內的有89個 $\Rightarrow \frac{\pi}{4} = 0.89 \quad \pi = 3.56$

產生1000個取樣點，落在圓內的有750個 $\Rightarrow \frac{\pi}{4} = 0.75 \quad \pi = 3$

由上面的例子可以得到一個結論，即觀察的取樣點越多，越接近原始的函數表現。

2.2 權重取樣

由 2.1 節我們已經知道積分可以用蒙地卡羅積分來近似，然而，並不是每一個機率分佈 $p(x)$ 都是適合取樣的。因此，我們可以假設一個很容易取樣的機率分佈 $q(x)$，並且由下式可以得知

$$\int f(x)p(x)dx = \int f(x)(\frac{p(x)}{q(x)})q(x)dx = E_{q(x)}[f(x)\frac{p(x)}{q(x)}] \cong \frac{1}{n}\sum_{i=1}^{n} f(x_i)\frac{p(x_i)}{q(x_i)} \qquad (2.4)$$

x_i 是由$q(x)$所產生的取樣點。另外一種表示方式爲

$\int f(x)q(x)dx \cong \frac{1}{n}\sum_{i=1}^{n} f(x_i)w(x_i),\ w(x) = \frac{p(x)}{q(x)}$ 被稱作每個取樣點的權重。而這樣的積分方法我們稱之爲權重取樣(Importance Sampling)。

2.3 馬可夫鏈

在介紹Metropolis-Hastings演算法與Gibbs sampler之前，先介紹一下馬可夫鏈的定義及其性質。定義 X_t 代表在時間 t 變數 x 的值(state)，所有變數 x 的可能值則形成一個集合，稱之爲值域(state space)。欲描述一個馬可夫鏈(Markov Chain)可以用其轉移機率(transition probabilities)來加以描述，列式如下

$$\Pr(X_{t+1} = s_{t+1} \mid X_0 = s_o, X_1 = s_1, ..., X_t = s_t) = \Pr(X_{t+1} = s_{t+1} \mid X_t = s_t) \qquad (2.5)$$

換句話說，馬克夫程序中的所有變數 x ，都只與所有變數 x 的可能值其前一個時間的值有關。如果一個序列中的每一個元素 $(X_0, ..., X_n)$ 都符合馬可夫程序之性質，則此序列可稱之爲馬可夫鏈。定義轉移機率 $P(i, j) = P(i \to j)$ ，表示一個馬可夫鏈由值 s_i 跳到 s_j 的機率，以式子來表示即爲

$P(i, j) = P(i \to j) = \Pr(X_{t+1} = s_j \mid X_t = s_i)$ 。我們以 $\pi_j(t) = \Pr(X_t = s_j)$ 來代表一個馬可夫鏈在時間t 時其值爲 j 的機率，$\pi(t)$ 代表在時間t 時所有值域中值的機率所形成的集合。以 $\pi(0)$ 代表一開始每個值的機率，通常假設爲等機率。在時間$t+1$ 的時候，值 s_i 的機率可以用*Chapman-Kolomogrov equation*來得到，如下所示：

$$\pi_i(t+1) = \Pr(X_{t+1} = s_i) = \sum_k \Pr(X_{t+1} = s_i \mid X_t = s_k) \cdot \Pr(X_t = s_k)$$
$$= \sum_k P(k \to i) \cdot \pi_k(t) = \sum_k P(k,i) \cdot \pi_k(t) \qquad (2.6)$$

Chapman-Kolomogrov equation描述了馬可夫鏈隨著時間的變化，也因此我們可以將其寫成矩陣的形式來分析。定義機率轉移矩陣(*probability transition matrix*) **P**，其第*i* 列第 *j* 行的元素*P*(*i*, *j*)代表了由值 *i* 到值 *j* 的機率，則Chapman-Kolomogrov equation可以寫成 $\pi(t+1) = \pi(t)\mathbf{P}$ 。

爲了描述一些馬可夫鍊的重要性質，我們先定義由值*i*經過n次轉換到值*j*的轉移機率 $p_{ij}^{(n)}$ ，換句話說， $p_{ij}^{(n)}$ 是 $\mathbf{P}^n$ 的第*i*列第*j*行的元素，即 $p_{ij}^{(n)} = \Pr(X_{t+n} = s_j \mid X_t = s_i)$ 。第一個性質是不可化簡性(irreducible)，意謂著所有的值都是可能出現的，其出現機率大於零，即 $p_{i,j}^{(n_{i,j})} > 0 \quad for\ all\ i, j$ 。如圖4所示。

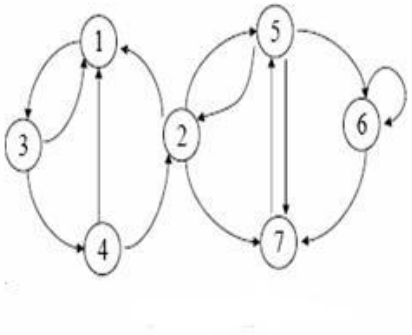


圖 4 具不可化簡性的馬可夫鏈

另外一個性質是非週期性，意謂著由值*i*到值*j* 所經過的轉換次數不會是某個數的整數倍。換句話說，當一個馬可夫鏈達到穩定的時候，代表每個值出現的機率不會再有所變動。我們以 π^* 來表示當馬可夫鏈達穩定時所有值的機率，並且很明顯地 π^* 與由哪個值開始無關，則平衡可以式子 $\pi^* = \pi^*\mathbf{P}$ 表示。要達成平衡的必要條件是此馬可夫鏈必須要是非週期並且不可化簡的。當一個馬可夫鏈是可化簡的時候，代表如果我們走到一個可以化簡的值時，就再也沒有辦法在有限的步驟中，抵達另外一個可能的值。當一個馬可夫鍊是週期性的，在其趨於穩定之後，其值出現的機率不會是固定的，而會在某些數字間交替週期性

的出現。馬可夫鏈是穩定的充分條件可以寫成細微平衡方程式(detailed balance function) ，即

$$p(j \to k)\pi_j^* = p(k \to j)\pi_k^* \text{ for } \forall i, j \text{ 。} \tag{2.7}$$

如果式(2.7)是成立的，則此馬可夫鏈又被稱爲可逆的(reversible)，因此式(2.7)又被稱爲可逆條件，因爲:

$$(\pi \mathbf{P})_j = \sum_i \pi_i p(i \to j) = \sum_i \pi_j p(j \to i) = \pi_j \sum_i p(j \to i) = \pi_j \tag{2.8}$$

2.4 Metropolis-Hastings Algorithm

在計算蒙地卡羅積分的時候，有一個很重要的議題就是「如何由複雜的機率分佈獲得取樣點」，而這也是馬可夫鍊-蒙地卡羅方法所要解決的最重要的問題。實際上，在20世紀中葉提出的Metropolis-Hasings algorithm (Metropolis and Ulam 1949, Metropolis et al. 1953, Hastings 1970)，目的就是爲了解決這個棘手的問題。由前人的研究[3]可知，一個馬可夫鏈的平衡機率 $\pi^*(.)$ 可以由下式表示

$$\pi^*(dy) = \int_{R_d} P(x, dy)\pi(x)dx \tag{2.9}$$

其中， $\pi(y)$ 爲 $\pi^*(.)$ 之Lebesque Measure，即 $\pi^*(dy) = \pi(y)dy$ ，且 $P(x, dy)$ 爲此馬可夫鏈的轉移機率且可以表示爲式(2.10)

$$P(x, dy) = p(x, y)dy + r(x)\delta_x(dy) \tag{2.10}$$

其中 $\delta_x(dy) = 1$ ，如果 $x \in dy$ ； $\delta_x(dy) = 0$ ，如果 $x \notin dy$ ，以及 $r(x) = 1 - \int_{R_d} p(x, y)dy$ 。

因此若能求出 $p(x, y)$ 即可知道一個馬可夫鏈的轉移機率。

舉例而言，假設一個任意的可能值 $x^{(0)}$ 與轉移機率 $q(x, y)$ ，由 $q(x, y)$ 產生一個取樣點 y ，若此 y 與 $x^{(0)}$ 間滿足細微平衡方程式，即

$$\pi(x^{(0)})q(y \mid x^{(0)}) = \pi(y)q(x^{(0)} \mid y) \tag{2.11}$$

則 y 即爲 $\pi(.)$ 的取樣點。當這樣的情形不成立時，例如當

$\pi(x^{(0)})q(y \mid x^{(0)}) > \pi(y)q(x^{(0)} \mid y)$ ，則我們需要定義一個比值 $\alpha(x, y)$ ，

使得 $\pi(x^{(0)})q(x^{(0)}, y)\alpha(x^{(0)}, y) = \pi(y)q(y, x^{(0)})$ 成立 。由前述可以知道 $\alpha(x, y)$ 可以

定義如下

$$\alpha(x,y)=\min\{1,\frac{\pi(y)p(y,x)}{\pi(x)p(x,y)}\} \tag{2.12}$$

茲將Metropolis-Hastings演算法整理如下

1.由任意可能的值 $x^{(0)}$ 出發，滿足 $p(x^{(0)})>0$ 。

2.j=1:N，由 $q(x^{(j)},.)$ 產生 y 及由一個均勻分配(Uniform Distribution)產生一個隨機變數 u 。

3.如果 $u\le\alpha(x^{(j)},y)$ ，則 $x^{(j+1)}=y$

4.若不符合則 $x^{(j+1)}=x^{(j)}$

接下來我們以圖5來說明M-H演算法

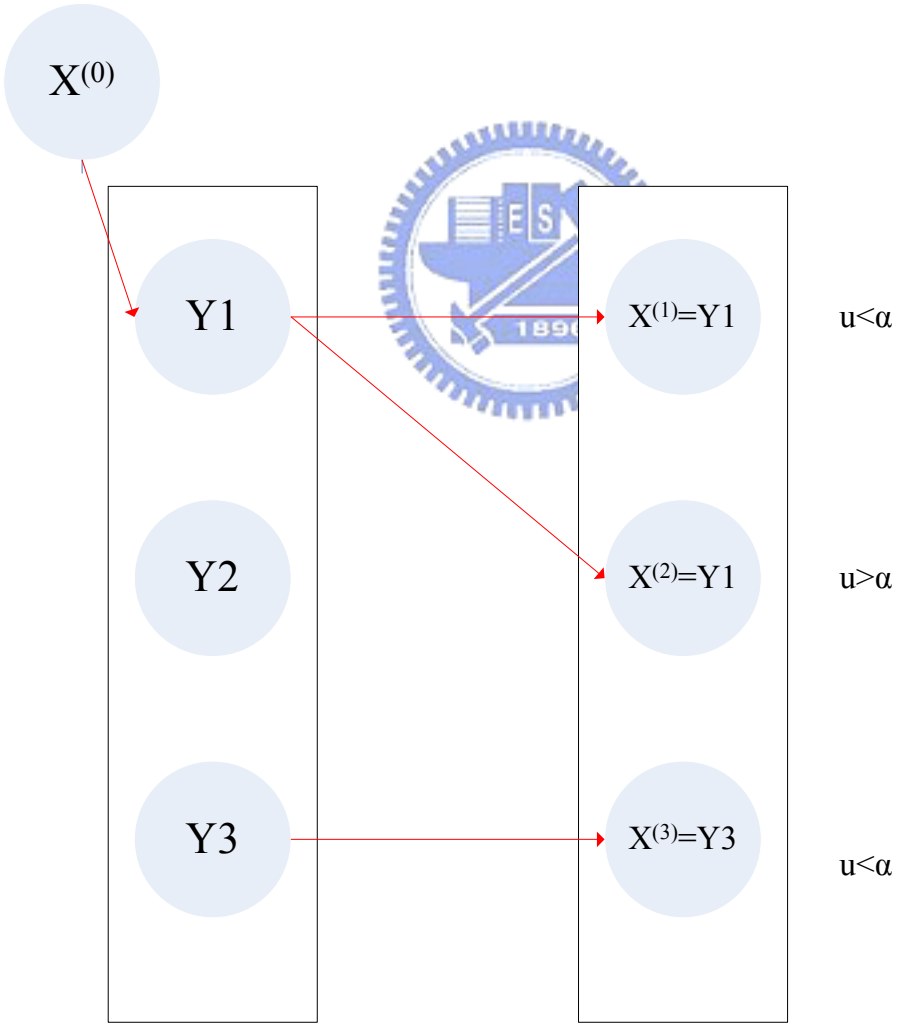


圖 5 馬可夫鏈的取樣變化過程

圖 5 中 Y_1 代表由 $q(X^{(0)},.)$ 所取樣的取樣點，而拒絕此取樣點的機率爲

$\alpha = \min\{1, \frac{\pi(Y_1)q(Y_1, X_0)}{\pi(X_0)q(X_0, Y_1)}\}$，在此假設$Y_1$被接受，即$X_1 = Y_1$。接下來我們重複上述動作，$Y_2$代表由$q(Y1,.)$所取樣的取樣點，但此時我們假設此取樣點被拒絕，此時$X^{(2)} = Y_1$。如此反覆進行，即可產生一組由$f(x)$所產生的取樣點$\mathbf{X}(t)$。

當重複幾次之後(稱爲訓練期間(burn-in period))，此馬可夫鍊會達到其平衡狀態，而在訓練期間之後所產生的取樣點即爲機率函數$\pi(.)$的取樣點。

2.5 The Gibbs Sampler

Gibbs sampler是由Geman在1984年所提出，原意是用來解決影像處理的問題，並且是Metropolis-Hastings取樣在接受機率永遠是一(即 α = 1)時的一個特例。Gibbs sampler的關鍵點在於將一個多維度的機率分佈視爲一個單維度的條件機率分佈，每一次觀察一個變數會比一次觀察多個變數來的簡單。換句話說，一個多隨機變數，我們可以將其中一個變數視爲未知，而假設其他的變數已知，並且給這些隨機變數一個初始值，在處理完一個變數之後，再選取下一個變數設爲未知，並且重複這個動作，即可將原本n維的機率分佈轉換成n個一維的機率分佈來處理。舉例來說，欲求一個二維的隨機機率分佈$p(x,y)$，但是只知道$p(x|y)$以及$p(y|x)$，假定由$p(x|y)$出發，任意給y一個初始值y_0，則我們可以由此機率分佈$p(x|y=y_0)$產生一個取樣點x_0，再將其帶回到$p(y|x)$，使用$p(y|x=x_0)$產生一個取樣點y_1，並且重複下去，即可產生一組(x,y)是由$p(x,y)$所產生的取樣點。寫成式子表示即爲

$x_i \sim p(x|y=y_{i-1}), i=1,2,.....n$

$y_i \sim p(y|x=x_{i-1}), i=1,2,.....n$

重複k次之後，產生了長度爲k的Gibbs sequence(x_j, y_j)，$1 \le j \le k$，這些(x_j, y_j)皆爲直接由$p(x,y)$做取樣的取樣點，也都具有$p(x,y)$的統計特性。最後，由Tierney在1994年的研究，我們可以知道Gibbs sequence最後會收斂到我們所想要的目標函數。以演算法形式表示如下

第一步：隨機產生一個 y_0

第二步： 令 $j=1$

第三步：對 $p(x_j \mid y_{j-1})$ 做取樣產生 x_j

第四步：對 $p(y_j \mid x_{j-1})$ 做取樣產生 x_j

之後迴圈執行到 $j=n$ 即停止。

Ex:

我們已知$f\left(x \mid y\right) \sim binomial(n, y)$

$$f(y \mid x) \sim beta(x+\alpha, n-x+\beta)$$

並已知x, y的聯合機率分佈函數

$$f(x \mid y) = C_x^n y^{x+\alpha-1}(1-y)^{n-x+\beta-1}$$

由直接計算我們可以直接推得隨機變數x的機率分佈爲

$$f(x) = C_x^n \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\alpha)} \frac{\Gamma(x+\alpha)\Gamma(n-x+\beta)}{\Gamma(\alpha+\beta+n)}$$

由圖6可以看出經由疊代所得到的變數x其機率分佈非常接近真正的機率分佈

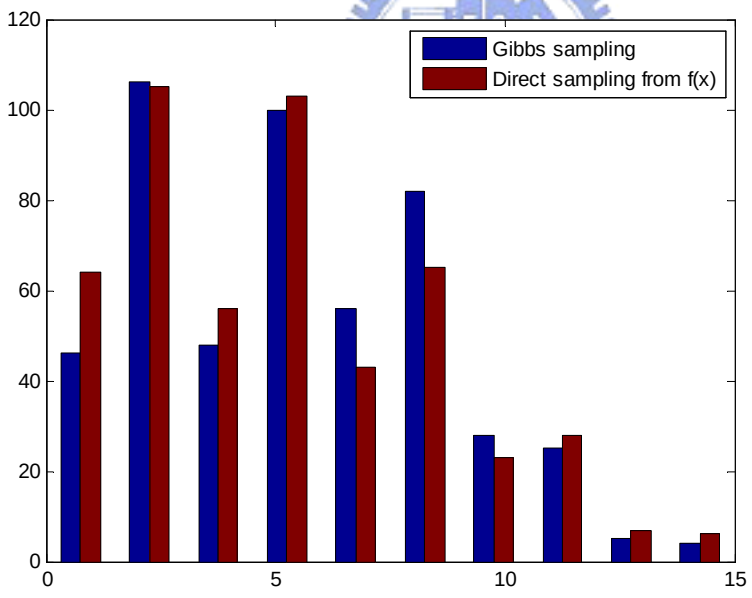


圖 6 取樣點數 500 點的機率分佈圖，其中α=2, β = 4 ,n=16,sample size=500

此外，當所需要取樣的變數超過兩個時，Gibbs sampler 一樣可以使用。假設隨機變數有$[x_1, x_2, ...x_d]$，並令其聯合機率密度函數(Joint Probability Density Function)爲 $f(x_1, x_2, ...x_d) = f(\mathbf{x})$。則我們將前面兩個變數的 Gibbs sampler 演算法修改如下：

第一步：隨機產生一個 $[x_1^{(0)}, x_2^{(0)}, ...x_d^{(0)}]$

第二步： 令 j=1

第三步：對 $f(x_1 \mid x_2^{(j-1)}, x_3^{(j-1)}, ...x_d^{(j-1)})$ 做取樣產生 $x_1^{(j)}$

對 $f(x_2 \mid x_1^{(j)}, x_3^{(j-1)}, ...x_d^{(j-1)})$ 做取樣產生 $x_2^{(j)}$

⋮

對 $f(x_d \mid x_2^{(j)}, x_3^{(j)}, ...x_{d-1}^{(j)})$ 做取樣產生 $x_d^{(j)}$

之後迴圈執行到 j = 需要的點數 爲止。

接下來，我們在此總結 Gibbs sampler 的優點如下：

1.不需要複雜的積分。

2.取樣機率分佈 q 可以任意選取。

3.不需要複雜的運算。

Gibbs sampler 的缺點：

1. q 的選擇。

2.起始值的選擇。

3.通常需要一段時間才能收斂。

第三章　軟式輸出維特比解碼理論(SOVA)

爲了更增進系統的效能，降低錯誤率，在現代的通訊系統中，都會使用錯誤更正技術，以對抗雜訊，增加通道的容量，其中最常被採用的手段就是錯誤更正碼(Forward Error Control Code)。錯誤更正碼即在我們所欲傳送的位元當中，加入一些預先設計好的多餘資訊，使得在接收端我們可以利用此多餘資訊將信號還原回來，這個過程我們稱爲通道編碼(Channel Coding)。通道編碼的方式可以分爲兩種，一種是區塊碼 (Block Code)，另一種則是摺積碼(Convolutional Code)。

在此處我們將專注於討論摺積碼。摺積碼最早由維特比(Andrew J. Viterbi)所提出，其主要概念是利用維特比理論(Viterbi Algorithm)來達成錯誤更正的目的。維特比理論[11]是由維特比在1967年所提出，其理論基礎在於利用格狀圖(Trellis Diagram)來搜尋所有可能的路徑並試著找出最有可能的那一條。藉由比較路徑所代表的輸入位元與接收到信號間的漢明距離(Hamming Distance)或是尤拉距離(Euclidean Distance)，距離最小者，機率最大，也因此可以得出我們所需要的解。此外，根據此種方法得出來的解爲最大概似解(Maximum Likelihood Solution)。

3.1　迴旋碼(Convolutional Code)

在傳送端我們利用簡單的移位暫存器(Shift Register)以及二位元加法器(Exclusive Or) 來進行摺積碼的編碼，以一個 $(5_8, 7_8)$ 的編碼器爲例：

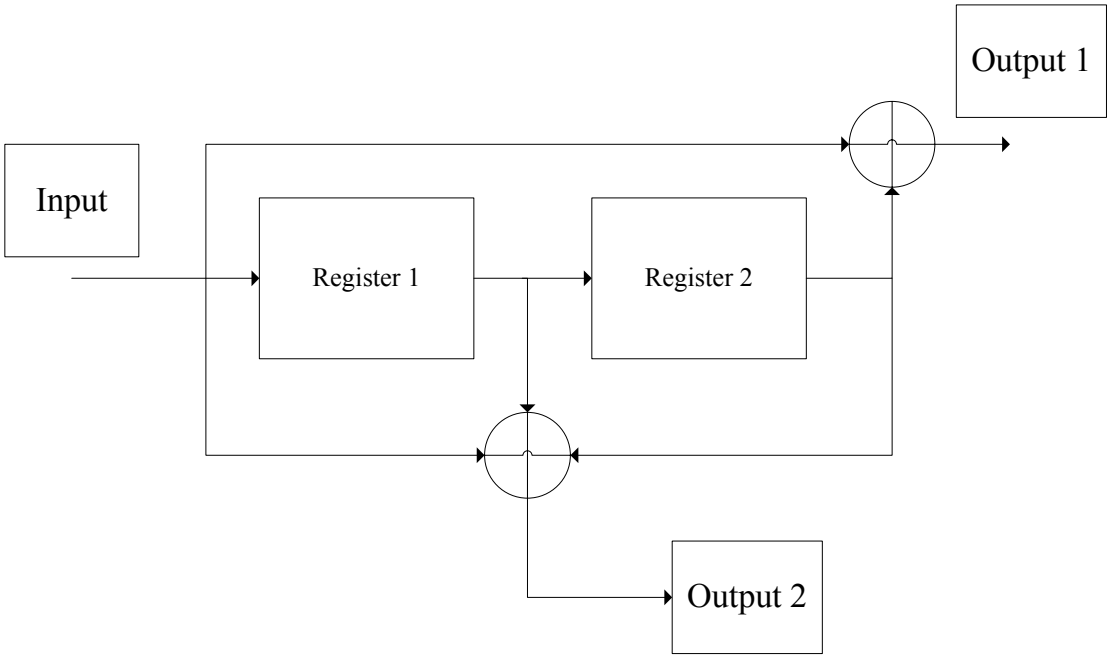


圖 7 一個(5,7)的迴旋碼編碼器

如上圖所示，每一個位元送入這個編碼器會得到兩個位元的輸出，也因此其編碼效率(Code Rate)爲1/2。其中 $(5_8, 7_8)$ 代表編碼多項式(Generator Polynomial)，當其以二進位方式 $(101_2, 111_2)$ 解讀時，可以描述哪些位移暫存器會輸出其內存值到二位元加法器進行運算。假設一個輸入的序列爲 $0,1,0,1,1,1,0,0,1,0,1,0,0,0,1_2$ ，則其輸出序列可以簡單地得出爲 00 ,11, 10, 00, 01, 10, 01, 11, 11, 10, 00, 10, 11, 00, 11_2 。底下我們將輸入，輸出的關係列成一張表

	輸出	
暫存器內存	**輸入 = 0:**	**輸入= 1:**
00	00	11
01	11	01
10	01	10
11	10	01

表格 1　迴旋積分碼編碼器輸入輸出關係圖

一般而言，如果原始資料的結尾不是00的話，在編碼時我們會在結尾多加上兩個位元00，，其用途在於將暫存器的狀態歸零。此種位元我們稱其爲尾端位元(tail bit)。

3.2 維特比理論(Viterbi Algorithm)

迴旋碼(Convolutional Code)的解碼端使用的是維特比理論[11]來進行解碼的動作，而最常用來說明解碼過程的圖稱之爲格狀圖(Trellis Diagram)。茲以上例爲例，我們可以畫出其格狀圖爲

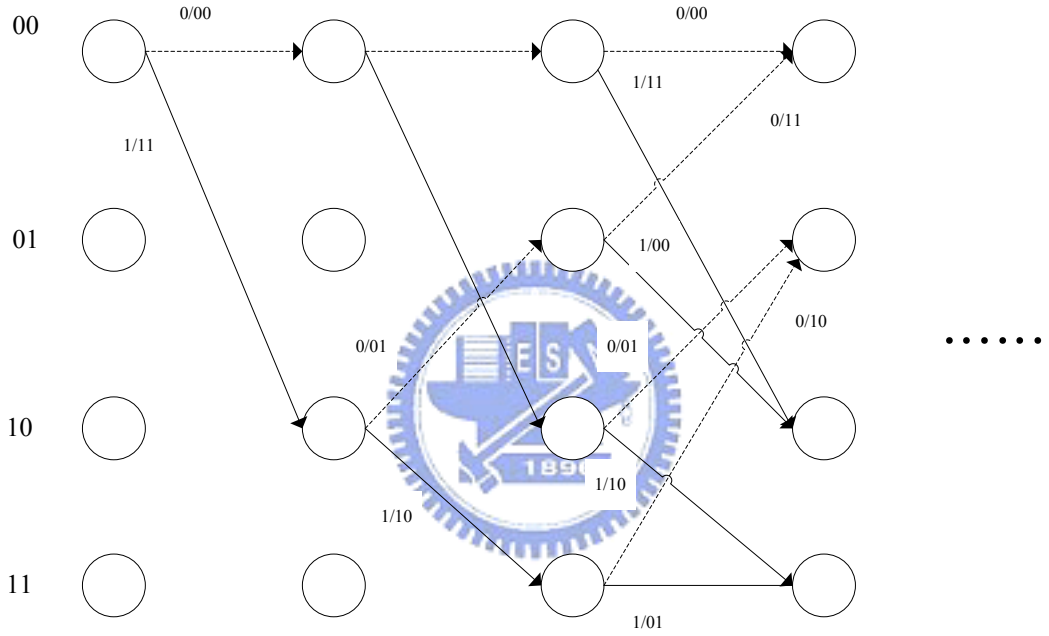


圖 8 (5,7)摺積碼之格狀圖

圖8中四個狀態(State)代表當時的暫存器狀態，而虛線代表輸出爲0的路徑(Path)，實線代表輸出爲1的分支。再更仔細地來看某一個特定區間的所有分支是如何轉換其狀態，就可以更清楚的了解維特比演算法，如下圖所示：

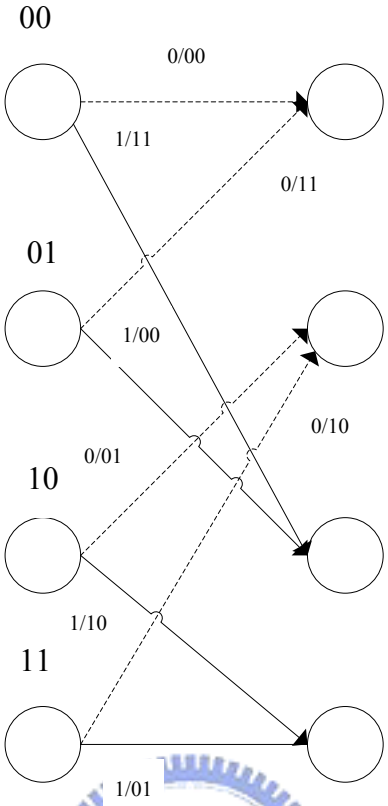


圖 9　格狀圖某一區間放大圖

根據我們對於接收信號的處理，可以將維特比演算法分爲硬式維特比演算法(Hard Decision Viterbi Algorithm)與軟式維特比演算法(Soft Decision Viterbi Algorithm)。硬式維特比演算法在剛接受到信號時會先做硬式決策(Hard Decision)，即所收到的信號大於0即判斷成+1，小於0則判斷成-1，之後再算其漢明距離。相對地，軟式維特比演算法並不會針對所收到的信號做任何處理，所運算的是其尤拉距離。舉例來說，假設在$t=0\sim t=1$之間，所收到的信號爲$(+0.8,-0.8)$，對硬式維特比演算法而言，會將其判定爲$(+1,-1)$，而其與由$state\ 00\rightarrow state\ 00$這個分支的路徑距離(Path Metric)可得爲$\sqrt{(1-(-1))^2+((-1)-(-1))^2}=2$。同樣的，對軟式維特比演算法而言，其距離爲$\sqrt{(0.8-(-1))^2+(-0.8-(-1))^2}\cong 1.811$。當有兩個分支同時到達同一個狀態(State)時，比較兩分支的大小，取其小者並記錄之。當格狀圖走到底時，我們可以藉由之前的紀錄得到一條距離最小的路徑，此路徑稱之爲存活路徑(Survivor Path)。此外，存活路徑所記錄的最小距離值，我們稱之

爲分支距離(Branch Metric)。值得注意的是，分支距離表示的是由起始點到觀察點的距離和，而路徑距離(Path metric)則表示某兩個觀察點間的距離。

3.3 軟式輸出維特比理論(Soft Output Viterbi Algorithm)

軟式輸出維特比理論(SOVA)是在1989年由Hagenauer[13]所提出，SOVA與前面所提到的軟式決策維特比演算法(Soft Decision Viterbi Algorithm)的差別在於：

1. 順向與逆向的路徑距離計算。
2. SOVA解碼器之輸出爲解碼後位元的後驗機率的LLR值(A Posteriori LLR)。

爲了等一下的說明，在這邊要先引入一個數學的觀念—Log Likelihood Ratio(LLR)，其定義爲

$$L(u_k) := \log(\frac{p(u_k=+1)}{p(u_k=-1)}) \tag{3.1}$$

其中，u_k 代表的是我們所傳送的那個位元，分子代表該位元爲1之機率，而分母則代表該位元爲0之機率。一般而言，若是在傳送端隨機地產生0與1，則這一項通常爲0。LLR值可以幫助我們決定該位元0與1的發生機率各有多少，其過程如下

$$\begin{aligned}
&e^{L(u_k)} = \frac{p(u_k=+1)}{1-p(u_k=-1)} \\
&p(u_k=+1)+p(u_k=-1)=1 \\
&\Rightarrow p(u_k=+1) = \frac{e^{L(u_k)}}{1+e^{L(u_k)}} = \frac{1}{1+e^{-L(u_k)}} \\
&\quad p(u_k=-1) = \frac{1}{1+e^{L(u_k)}} = \frac{e^{-L(u_k)}}{1+e^{-L(u_k)}} \\
&\Rightarrow p(u_k=\pm 1) = (\frac{e^{-\frac{L(u_k)}{2}}}{1+e^{-L(u_k)}})e^{\pm\frac{L(u_k)}{2}}
\end{aligned} \tag{3.2}$$

根據後驗機率所求出的比值，我們稱之爲條件的LLR值(Conditional LLR)，其定義爲 $L(u_k \mid y) := \log(\frac{p(u_k=+1 \mid y)}{p(u_k=-1 \mid y)})$。舉例而言，對於一個常見的白高斯雜訊(AWGN)通道，其傳送信號x=1之事後機率分佈爲 $p(y \mid x=+1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-1)^2}{2\sigma^2}}$ ；同理，x=0

之事後機率分佈爲 $p(y \mid x=-1)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(y+1)^2}{2\sigma^2}}$ 其條件LLR值可以經由式(3.1)計算而得

$$\begin{aligned} L(u_k \mid y) &= \log\left(\frac{p(u_k=+1 \mid y)}{p(u_k=-1 \mid y)}\right) \\ &= \log\left(\frac{p(y \mid u_k=+1)}{p(y \mid u_k=-1)}\right) \\ &= \log\left(e^{-\frac{1}{2\sigma^2}[-(y-1)^2-(y+1)^2]}\right) \\ &= \frac{2}{\sigma^2}y = L_c y \end{aligned} \tag{3.3}$$

式(3.3)中的第二個等號是根據貝式定理所推導得出的結果，其中 L_c 又稱做可靠度值(Reliablity Value)，且只跟信號雜訊比(Signal to Noise Ratio, SNR)有關。又根據貝式定理，存在 $p(u_k \mid y)=p(y \mid u_k)p(u_k)$ 這樣的關係，帶入式(3.1)可以導出事前機率與後驗機率的關係爲

$$L(u_k \mid y)=L(u_k)+L(y \mid u_k) \tag{3.4}$$

其中， $L(u_k)$ 對應式(3.1)，因此項通常爲事前已知的機率，故又被稱爲事前資訊(A Prior Information)或是本質資訊(Intrinsic Information)； $L(y \mid u_k)$ 爲事後機率的比值，又常被稱作外部資訊(Extrinsic Information) ； $p(u_k \mid y)$ 一般被稱做後驗機率(A Posteriori Probability)，因此 $L(u_k \mid y)$ 常被稱爲後驗資訊(A Posteriori Information)。

接下來，爲了簡化，我們定義一個系統，此系統使用一個碼率 (Code Rate)爲1/N的編碼，且每一個端點(node)只有兩條分支會同時抵達，在經過一段時間 δ 後，這段時間必須足夠長以便所有可能的存活路徑會合爲一條。 ν 代表的是暫存器的個數，同時狀態個數也可以由 $S=2^{\nu}$ 來求出。如下圖所示

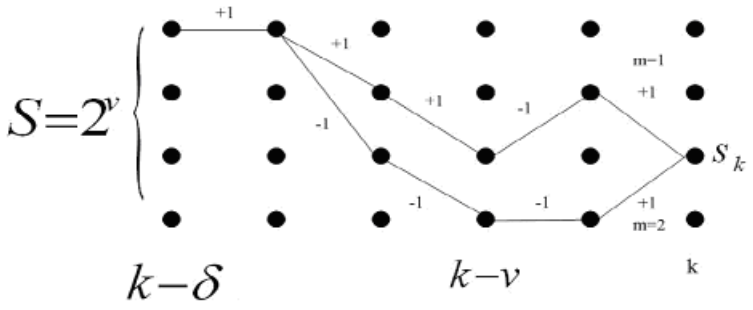


圖 10 SOVA 示意圖[摘錄自 reference 13]

此時，傳統的維特比演算法必須決定一條最有可能的路徑，而其決定的方法是選擇具有最小距離(Metric)的路徑。以AWGN爲例[13]

$$M_m = \frac{E_b}{N_o} \sum_{j=k-\delta}^{k} \sum_{n=1}^{N} (y_{jn} - x_{jn}^{(m)})^2 \qquad m = 1,2 \tag{3.5}$$

$x_{jn}^{(m)}$ 代表的是在時間 j 時，第 m 條路徑上的總共 N 個位元中的第 n 個位元，y_{jn} 則代表相同時間所收到的第 n 個位元，且 $\frac{E_b}{N_o}$ 爲信號雜訊比(Signal to Noise Ratio)。

由這樣的系統我們可以得到，選擇第 m 個路徑的機率爲

$$prob(path\ m) \approx e^{-M_m} \qquad m = 1,2 \tag{3.6}$$

此外，我們假設比較小的距離爲 M_1，即 $M_1 \le M_2$。換句話說，選到錯誤路徑(即選到 M_2)的機率爲

$$p_w = \frac{e^{-M_2}}{e^{-M_1} + e^{-M_2}} = \frac{1}{1 + e^{M_2 - M_1}} = \frac{1}{1 + e^{\Delta}} \tag{3.7}$$

其中，$\Delta = M_2 - M_1 \ge 0$。這樣的一個系統，維特比演算法所犯的錯誤即爲第 2 條路徑與第 1 條路徑的不同處。假設有 e 個地方不同，即 $u_j^{(1)} \ne u_j^{(2)} \quad j = 1,2,.....e$。

此時，若路徑 1 在第 j 個錯誤之前累積之錯誤機率爲 $\hat{p}_j$，在路徑 1 被選取的前提下，第 j+1 個錯誤的發生機率爲

$$\hat{p}_j \leftarrow \hat{p}_j(1 - p_w) + (1 - \hat{p}_j)p_w \qquad j = 1,2,.....e \tag{3.8}$$

注意式(3.8)成立的條件爲 p_w 與 $\hat{p}_j$ 兩者獨立，而在大部份的編碼下這都是成立的。

此時每個位元的 LLR 可以求得

$$\hat{L}_j = \log(\frac{1-\hat{p}_j}{\hat{p}_j}) \qquad 0 \le \hat{L}_j \le \infty \tag{3.9}$$

由(3.7)(3.8)(3.9)可以推得

$$\hat{L}_j \leftarrow f(\hat{L}_j, \Delta) = \frac{1}{\alpha} log(\frac{1+e^{\alpha \hat{L}_j + \Delta}}{e^{\Delta} + e^{\alpha \hat{L}_j}}) \tag{3.10}$$

其中， α 爲一常數，其目的在於使得 $E[\hat{L}_j] = 1$ 。

一個適當的 α 值可以選取爲

$$\alpha = 4 d_{free} \frac{E_s}{N_o} \tag{3.11}$$

又，式(3.10)有一個良好的近似如下

$$f(\hat{L}_j, \Delta) = \min\{\hat{L}_j, \frac{\Delta}{\alpha}\} \tag{3.12}$$

接下來我們將此演算法做一個整理。

儲存部分：

k: time index(modulo δ+1)

$\hat{\mathbf{u}}(s_k) = [\hat{u}_{k-\delta}(s_k), ... \hat{u}_k(s_k)]$

{ 硬式決策值 $\hat{u} \in \pm 1$ }

$\hat{L}(s_k) = [\hat{L}_{k-\delta}(s_k), ... \hat{L}_k(s_k)]$

{ 軟式資訊值 }

$\Gamma(s_k)$: 累積距離值(*Accumulated metrice value*)

迴圈部分：

a)傳統Viterbi algorithm

計算 $\Gamma(s_{k-1}, s_k) = \Gamma(s_{k-1}) + \frac{E_s}{N_o} \sum_{n=1}^{N} (y_{kn} - x_{kn}^{(m)})^2$

找到$\Gamma(s_k)$的最小值儲存並且存取相對應的存活者 $\hat{u}_k(s_k)$

b)決定軟式資訊

每個狀態(state)s_k其軟式資訊爲

$\Delta = \max \Gamma(s_{k-1}, s_k) - \min \Gamma(s_{k-1}, s_k)$

$j = k - \nu \ to \ j = k - \delta$

比較進入狀態s_k的兩個路徑

如果$u_j^1(s_j) \neq u_j^2(s_j)$

則$\hat{L}_j := f(\hat{L}_j, \Delta)$

第四章　球狀解碼

4.1 球狀解碼

在通訊系統中，球狀解碼(Sphere Decoding)是一種常被用來尋找最大近似解(Maximum-Likelihood Solution)的方法，其所欲解之最基本的方程式如下：

$$\hat{\mathbf{X}}_{ML} = \arg\min_{\mathbf{X}\in\Lambda} \|\mathbf{Y}-\mathbf{A}\mathbf{X}\|^2 \tag{4.1}$$

$\mathbf{A}$ 指的是一個 M 列 N 行的矩陣，此處假設 $M < N$ ，$\mathbf{Y}$ 是一個 N 列1行的矩陣，$\mathbf{X}$ 亦是 N 列 1 行的矩陣，Λ 則爲所描述的是所有可能解 $\mathbf{X}$ 所形成之集合。有兩種方法較常用來解式(4.1)的問題，其中一種是利用 Cholosky 分解，其過程如下所示：

$$\begin{aligned}
&\hat{\mathbf{X}}_{ML} = \arg\min_{\mathbf{X}\in\Lambda} \|\mathbf{Y}-\mathbf{A}\mathbf{X}\|^2 \\
&= \arg\min_{\mathbf{X}\in\Lambda} (\mathbf{Y}-\mathbf{A}\mathbf{X})^T(\mathbf{Y}-\mathbf{A}\mathbf{X}) \\
&= \arg\min_{\mathbf{X}\in\Lambda} \mathbf{Y}^T\mathbf{Y}-\mathbf{Y}^T\mathbf{A}\mathbf{X}-\mathbf{X}^T\mathbf{A}^T\mathbf{Y}+\mathbf{X}^T\mathbf{A}^T\mathbf{A}\mathbf{X} \\
&= \arg\min_{\mathbf{X}\in\Lambda} (\mathbf{Y}^T\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}-\mathbf{Y}^T\mathbf{A}\mathbf{X}-\mathbf{X}^T\mathbf{A}^T\mathbf{Y}+\mathbf{X}^T\mathbf{A}^T\mathbf{A}\mathbf{X})+[\mathbf{Y}^T\mathbf{Y}-\mathbf{Y}^T\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}] \\
&= \arg\min_{\mathbf{X}\in\Lambda} (\mathbf{Y}^T\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{I}\mathbf{A}^T\mathbf{Y}-\mathbf{Y}^T\mathbf{A}\mathbf{I}\mathbf{X}-\mathbf{X}^T\mathbf{I}\mathbf{A}^T\mathbf{Y}+\mathbf{X}^T\mathbf{A}^T\mathbf{A}\mathbf{X})+\mathbf{Y}^T(\mathbf{I}-\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T)\mathbf{Y} \\
&= \arg\min_{\mathbf{X}\in\Lambda} \mathbf{Y}^T\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}(\mathbf{A}^T\mathbf{A})(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}-\mathbf{Y}^T\mathbf{A}(\mathbf{A}^T\mathbf{A})(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{X}-\mathbf{X}^T(\mathbf{A}^T\mathbf{A})(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y} \\
&\qquad +\mathbf{X}^T\mathbf{A}^T\mathbf{A}\mathbf{X})+\mathbf{Y}^T(\mathbf{I}-\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T)\mathbf{Y} \\
&= \arg\min_{\mathbf{X}\in\Lambda} \mathbf{Y}^T(\mathbf{A}^\dagger)^T\mathbf{A}^T\mathbf{A}\mathbf{A}^\dagger\mathbf{Y}-\mathbf{Y}^T(\mathbf{A}^\dagger)^T\mathbf{A}^T\mathbf{A}\mathbf{X}-\mathbf{X}^T\mathbf{A}^T\mathbf{A}\mathbf{A}^\dagger\mathbf{Y}+\mathbf{X}^T\mathbf{A}^T\mathbf{A}\mathbf{X}) \\
&\qquad +\mathbf{Y}^T(\mathbf{I}-\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T)\mathbf{Y} \\
&= \arg\min_{\mathbf{X}\in\Lambda} (\mathbf{A}(\mathbf{A}^\dagger\mathbf{Y}-\mathbf{X}))^T(\mathbf{A}(\mathbf{A}^\dagger\mathbf{Y}-\mathbf{X}))+\mathbf{Y}^T(\mathbf{I}-\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T)\mathbf{Y} \\
&= \arg\min_{\mathbf{X}\in\Lambda} (\hat{\mathbf{X}}-\mathbf{X})^T\mathbf{A}^T\mathbf{A}(\hat{\mathbf{X}}-\mathbf{X})+\beta
\end{aligned} \tag{4.2}$$

在第(4.2)式中，β 代表的是一個常數，並且不隨 $\mathbf{X}$ 而改變。$\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ 代表 $\mathbf{A}$ 之假反矩陣(Pseudo-Inverse)，即 $\hat{\mathbf{X}}$ 代表的是 $\mathbf{A}\mathbf{X}=\mathbf{Y}$ 之最小平方近似解(Least Square Solution)。換言之，欲解第(4.1)式，即等效於解第(4.3)式如下：

$$\hat{\mathbf{X}}_{ML} = \arg\min_{\mathbf{X}\in\Lambda} (\hat{\mathbf{X}}-\mathbf{X})\mathbf{A}^T\mathbf{A}(\hat{\mathbf{X}}-\mathbf{X}) \tag{4.3}$$

一個最簡單的方法就是嘗試 Λ 中每個可能的解 $\mathbf{X}$ 直到找到滿足上式的最小解爲止，這種方法我們稱爲窮舉搜尋法(Exhaustive Search)。然而，窮舉搜尋法的複雜度太高以至於非常難以實現。也因此，球狀解碼(Sphere Decoding)可以解決這個問題，其方法的概念爲訂定一個搜尋半徑並在其中尋找解答，如圖 11 所示。明顯地，以這種概念來尋找可以避免嘗試過多的解，且最大近似解，必定會落在此球當中。以式(4.4)表示即爲：

$$\hat{\mathbf{X}}_{ML} = \arg\min_{\mathbf{X}\in\Lambda}(\hat{\mathbf{X}}-\mathbf{X})\mathbf{A}^T\mathbf{A}(\hat{\mathbf{X}}-\mathbf{X}) \le r^2 \tag{4.4}$$

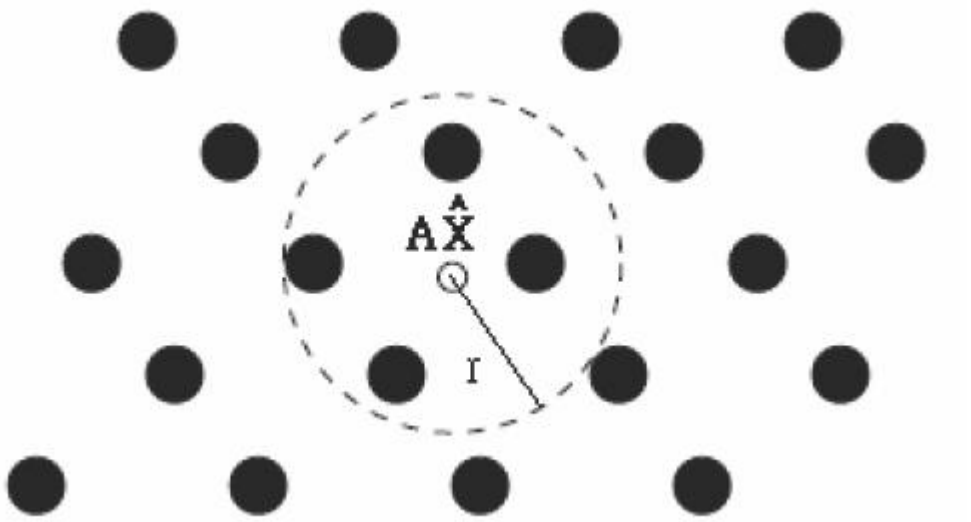


圖 11 球狀解碼示意圖 [摘錄自 reference 15]

若是我們可以求得一個上三角矩陣 $\mathbf{U}$ ，使其滿足 $\mathbf{U^TU} = \mathbf{A^TA}$ ，則式(4.4)可以改寫成

$$\begin{aligned}(\hat{\mathbf{X}}-\mathbf{X})\mathbf{A}^T\mathbf{A}(\hat{\mathbf{X}}-\mathbf{X}) &= (\hat{\mathbf{X}}-\mathbf{X})\mathbf{U}^T\mathbf{U}(\hat{\mathbf{X}}-\mathbf{X}) \\ &= \sum_{i=1}^{M} u_{ii}^2[x_i - \hat{x}_i + \sum_{j=i+1}^{M}\frac{u_{ij}}{u_{ii}}(x_j - \hat{x}_j)]^2\end{aligned} \tag{4.5}$$

由 $i = M$ 開始可得

$$u_{MM}^2(x_M - \hat{x}_M)^2 \le r^2 \tag{4.6}$$

並且將 $i = M, M-1, \ldots 1$ 依次代入。

然而，此處會產生兩個問題：

1. 如何決定半徑 r ？如果選的太大，則球內之可能解太多，運算複雜度仍然很

高；如果選的太小，則球內可能不存在可能解。

2. 如何決定某個點是否存在球中？若是我們直接求每個可能解與收到信號間的距離，則仍然需要做 N 維的運算。

球狀解碼並沒有明確地回答第一個問題，但是對於第二個問題，確實有提出一個非常有效率的解法。要如何解決第二個問題呢？直覺地，解決一個一維空間的問題是簡單的，而解決一個 N 維空間的問題則非常複雜。因此，若是我們能將一個 N 維空間的問題拆解成 N 個一維空間的問題，複雜度將可以得到明顯的降低。舉例而言，假設一個集合，其中的元素是滿足所有 K 個維度都落在我們預設的搜尋球中的點，則我們在尋找可能解時，只需要觀察在這樣的一個集合當中，哪一個解的第 K+1 個維度也落在此球中。原因在於若是前面 K 個維度就已經不存在搜尋球中，則第(K+1)的維度當然亦不可能在搜尋範圍當中。至於第一個問題，在此我們使用最小平方近似解(Least Square Solution)來決定球之半徑，原因是最小平方近似解爲在沒有任何限制條件之下，所能得出的最佳解。且以如此的搜尋法，最少會存在一個解在我們的搜尋球當中，這個解就是我們的最小平方近似解。最小平方近似解可以由式子 $\hat{\mathbf{X}}=(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}$ 求出。

另外一種方法，我們可以針對矩陣 $\mathbf{A}$ 做 QR 分解，以式子表示如下：

$$\mathbf{A}=\mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}_{(N-M)xM}\end{bmatrix} \tag{4.7}$$

其中 $\mathbf{R}$ 爲 $M\times M$ 的上三角矩陣，$\mathbf{Q}=[\mathbf{Q}_1\ \ \mathbf{Q}_2]$ 代表正交矩陣 Q 的前面 M 行與後面(N-M)行。 由(4.5)可以再往下推得

$$\begin{aligned} d^2 &\geq \left\|\mathbf{Y}-[\mathbf{Q}_1\ \ \mathbf{Q}_2]\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}\mathbf{X}\right\|^2 \\ &= \left\|\begin{bmatrix}\mathbf{Q}_1^*\\ \mathbf{Q}_2^*\end{bmatrix}\mathbf{Y}-\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}\mathbf{X}\right\|^2 \\ &= \left\|\mathbf{Q}_1^*\mathbf{Y}-\mathbf{R}\mathbf{X}\right\|^2+\left\|\mathbf{Q}_2^*\mathbf{Y}\right\|^2 \end{aligned} \tag{4.8}$$

其中 $()^*$ 代表的是做赫密特(Hermitian)矩陣運算。由上式可以繼續推得

$$d^2-\left\|\mathbf{Q}_2^*\mathbf{Y}\right\|^2=\left\|\mathbf{Q}_1^*\mathbf{Y}-\mathbf{R}\mathbf{X}\right\|^2 \tag{4.9}$$

接下來我們令 $\mathbf{Z}=\mathbf{Q}_1^*\mathbf{Y}$ 以及 $d'^2=d^2-\left\|\mathbf{Q}_2^*\mathbf{Y}\right\|$，則式(4.7)可以被改寫成爲

$$d'^2 \geq \sum_{i=1}^{M}(z_i-\sum_{j=1}^{M}r_{i,j}x_j)^2 \tag{4.10}$$

其中 $r_{i,j}$ 代表在矩陣 $\mathbf{R}$ 當中的元素，z_i 代表矩陣 $\mathbf{Z}$ 中的元素，而 x_j 則代表了矩陣 $\mathbf{X}$ 中的元素。爲了更進一步地了解式(4.8)，我們將式(4.8)中的 $\sum$ 符號展開，可以得到

$$d'^2 \geq (z_M-r_{M,M}x_M)^2+(z_{M-1}-r_{M-1,M}x_M-r_{M-1,M-1}x_{M-1})^2+\cdots \tag{4.11}$$

由式(4.9)可以得知，第一項僅跟 x_M 有關；換句話說，此解 $\mathbf{X}$ 落在球中的必要條件爲

$$d'^2 \geq (z_M-r_{M,M}x_M)^2 \tag{4.12}$$

藉由 QR 分解，我們將原本是 M 維的向量問題，分解爲一次只考慮一維的向量問題。式(4.10)的另外一種表示方法爲

$$\left\lceil \frac{-d'+z_M}{r_{M,M}} \right\rceil \leq s_M \leq \left\lfloor \frac{d'+z_M}{r_{M,M}} \right\rfloor \tag{4.13}$$

在此 $\lfloor . \rfloor$ 代表最靠近且最小的可能解，$\lceil . \rceil$ 代表最靠近且最大的可能解。找出所有符合的 s_M 後，我們只需要針對這些 s_M 來檢查他的下一個維度是否滿足式(4.9)，即令 $d'^2_{M-1}=d'^2-(z_M-r_{M,M}x_M)^2$ (4.12)以及 $z_{M-1|M}=z_{M-1}-r_{M-1,M}x_M$ 。經過這樣的假設之後，我們可以將式(4.11)往上推一個維度到 M-1，並且決定 x_{M-1} 的範圍。茲以式(4.12)表示如下：

$$\left\lceil \frac{-d_{M-1}'+z_{M-1|M}}{r_{M-1,M-1}} \right\rceil \leq s_{M-1} \leq \left\lfloor \frac{d_{M-1}'+z_{M-1|M}}{r_{M-1,M-1}} \right\rfloor \tag{4.14}$$

重複如上之動作，即可求出信號 M 個維度的解，而這樣的蒐尋過程恰巧可以樹狀圖的方式加以說明。茲繪出一個可能的樹狀圖範例如下：

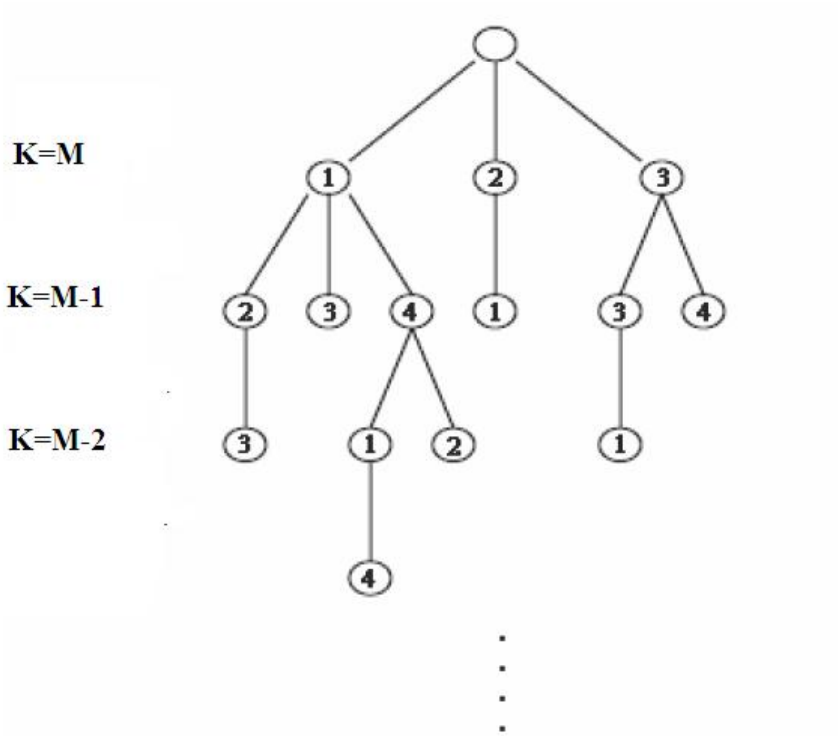


圖 12 一個可能的樹狀圖結構 [摘錄自 reference 15]

圖 12 中，左邊的 K 代表我們解到信號的第 K 個維度，而圓圈中的數字代表的是 x_K 有幾個可能的解(以本圖爲例，一個 QPSK 的系統其解之最多可能有四個)。

4.2 複數球狀解碼

在 4.1 節中所描述的，爲考慮實數系統下之球狀解碼。然而，實數系統僅能運用在二相位鍵移(Binary Phase Shift Keying)調變上，當我們想進一步提升傳輸速率時，勢必得使用更複雜的條變，例如四相位鍵移(Quadrature Phase Shift Keying)。其差別在於式(4.3)中的 $\mathbf{A}$ 、 $\mathbf{X}$ 與 $\hat{\mathbf{X}}$ 都是複數。有兩種方法可以解決這樣的問題；第一種是將 $\mathbf{A}$ 、 $\mathbf{X}$ 與 $\hat{\mathbf{X}}$ 分成實數的部分與虛數的部分，如此這兩個部分都可以被視爲獨立的實數系統。我們將式子(4.3)改寫如下

$$\tilde{\mathbf{X}}_{ML} = \arg\min(\tilde{\mathbf{X}}_{LS} - \tilde{\mathbf{X}})^T \tilde{\mathbf{A}}(\tilde{\mathbf{X}}_{LS} - \tilde{\mathbf{X}}) \tag{4.15}$$

其中

$$\tilde{\mathbf{X}} = [\mathrm{Re}\{\mathbf{X}^T\} \quad -\mathrm{Im}\{\mathbf{X}^T\}]$$

$$\tilde{\mathbf{X}}_{LS} = [\mathrm{Re}\{\hat{\mathbf{X}}^T\} - \mathrm{Im}\{\hat{\mathbf{X}}^T\}]$$

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathrm{Re}\{\mathbf{A}^T\} & -\mathrm{Im}\{\mathbf{A}^T\} \\ \mathrm{Im}\{\mathbf{A}^T\} & \mathrm{Re}\{\mathbf{A}^T\} \end{bmatrix}$$

$$\tilde{\Lambda} = [\{\mathrm{Re}(\Lambda) \qquad \mathrm{Im}(\Lambda)\}]$$

如果 $\mathbf{X}$ 是一個 QPSK 系統，則 $\tilde{\mathbf{X}}$ 爲兩個 BPSK 系統(包含實部與虛部)。令 x_M 是向量 $\mathbf{X}$ 之第 M 個維度，且 $x_M = r_c e^{j\theta_M}$ ，其中 $\theta_M \in \{0, \frac{2\pi}{2^{M_c}}, \ldots\ldots \frac{2\pi(2^{M_c}-1)}{2^{M_c}}\}$ 爲 2^{M_c} 角度相鍵移的星座圖(Constellation)， r_c 爲形成此星座圖之半徑，爲一個信號(Symbol)所代表的符元(bit)數，如下兩圖所示：

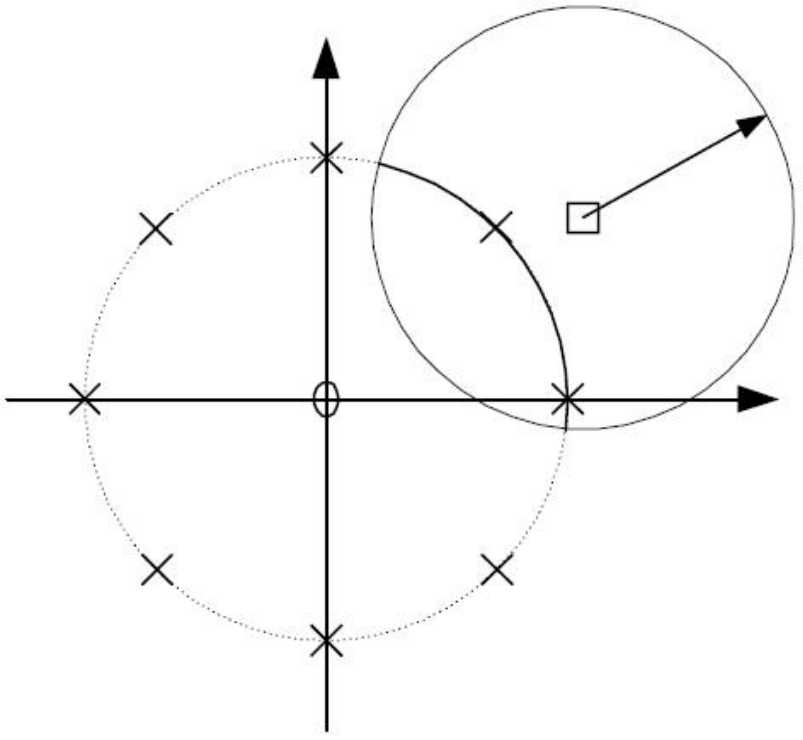


圖 13 8PSK 複數信號搜尋半徑[摘錄自 reference 17]

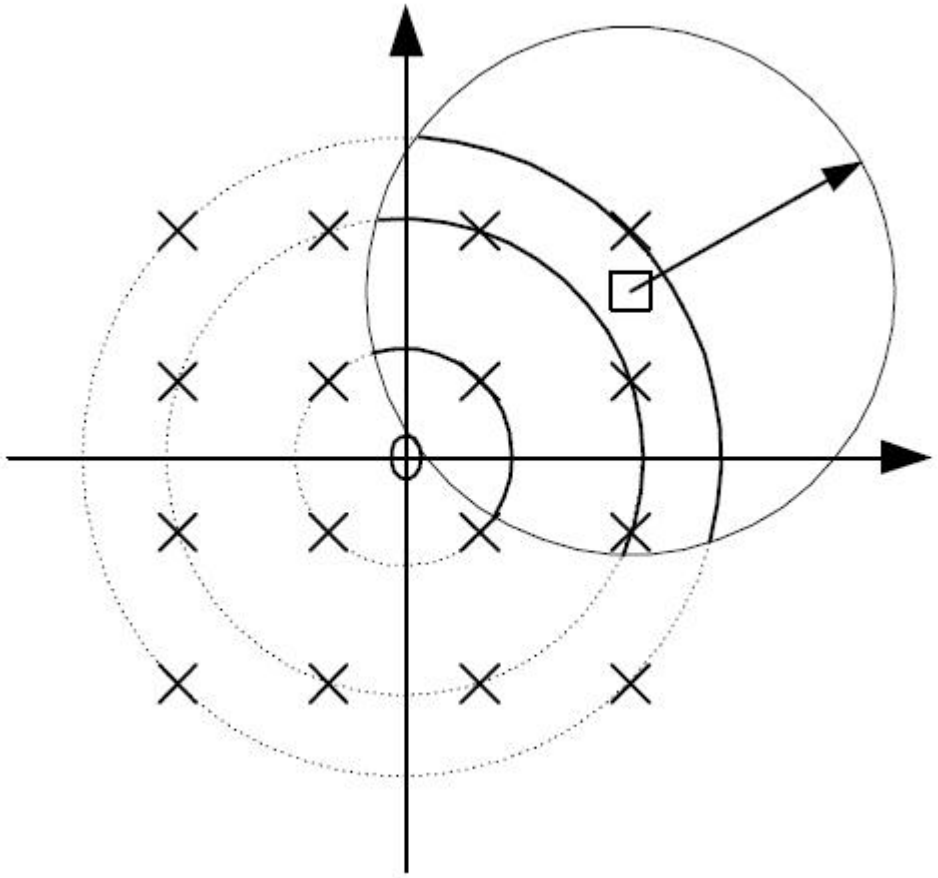


圖 14　16QAM 複數信號搜尋半徑圖[摘錄自 reference 17]

因爲是複數系統，所以將 $x_M = r_c e^{j\theta_M}$ 帶入式(4.10)可以得到如下列之結果：

$$x_M = r_c e^{j\theta_M} \quad z_M = \hat{r}_c e^{j\hat{\theta}_M}$$

$$\left\| z_M - r_{M,M} x_M \right\|^2 = r_c^2 + r_{M,M}^2 \hat{r}_c^2 - 2 r_c r_{M,M} \hat{r}_c \cos(\theta_M - \hat{\theta}_M) \le d'^2$$

$$\cos(\theta_M - \hat{\theta}_M) \le \frac{d'^2 - r_c^2 - r_{M,M}^2 \hat{r}_c^2}{2 r_c r_{M,M} \hat{r}_c} := \eta \tag{4.16}$$

$$\Rightarrow \left\lceil \frac{2^{M_c}}{2\pi}(\hat{\theta}_M - \cos^{-1}(\eta)) \right\rceil \le \frac{2^{M_c}}{2\pi}\theta_M \le \left\lfloor \frac{2^{M_c}}{2\pi}(\hat{\theta}_M + \cos^{-1}(\eta)) \right\rfloor \tag{4.17}$$

對於 M-QAM 的系統而言，因爲不同的可能解其 r_c 皆不同，因此當我們要搜尋落在此球中的可能點時，僅需解式(4.15)，滿足式(4.15)即滿足式(4.10)[ref]。

4.3　表列式球狀解碼(List Sphere Decoding)

表列式球狀解碼是前述之球狀解碼的一種變形，當我們採用表列式球狀解碼

時，解碼器會產生一個表，其內容爲所有滿足式(4.1)條件的可能解 **X** 。若我們欲將球狀解碼修正爲表列式球狀解碼，我們需要實行兩個步驟：

1. 每找完一個維度不要將半徑更新，即式(4.12)在表列式球狀解碼中並不會被執行。如此可以避免我們遺漏了某些可能的解。

2. 表所預定的空間大小爲 L。若是 L 未滿則將可能解存入；若是已經沒有空間，則將所儲存的可能解中距離所接收到的信號最遠的那一個解丟棄，並將新的可能解存入。

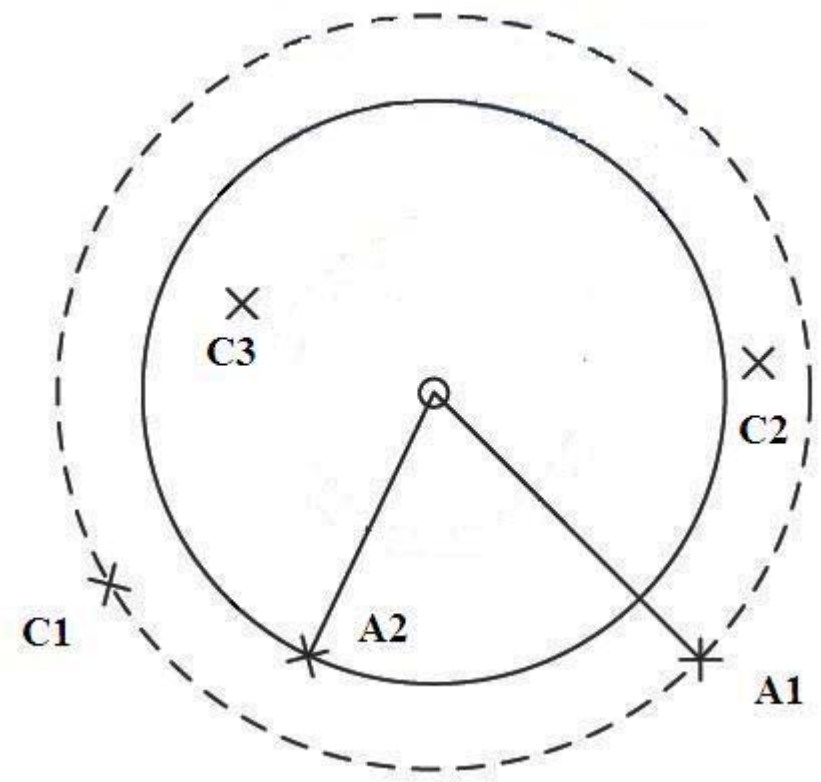


圖 15 表列式球狀解碼與傳統球狀解碼之半徑比較圖
[摘錄自 reference 15]

由上圖可以很清楚地解釋表列式球狀解碼與球狀解碼的不同。圖中圓心到 A1 代表的是表列式球狀解碼的半徑，而圓心到 A2 代表的是球狀解碼的半徑。由圖中可以很明顯地知道，C1 以及 C2 是落在表列式球狀解碼的可能解範圍中，若是單純的球狀解碼則會將此兩個解排除掉。C3 則是對兩種方式而言都會是可能解。

第五章　渦輪式解碼器(Turbo Decoder)

5.1 使用馬可夫鏈-蒙地卡羅方法的渦輪式解碼器

在這一章中，我們所要描述一個軟式信號決策(Soft Decision)的系統，並且說明前幾章所提到的內容如何實做一個軟進-軟出(Soft-in Soft-out)的系統。欲說明一個這樣的過程，我們先定義我們所採用的多輸入-多輸出(Multi-Input Multi-Output)的通道模型。茲列如下：

$$\mathbf{y} = \mathbf{Hd} + \mathbf{n} \tag{5.1}$$

其中，**y** 代表我們所收到的信號，矩陣大小爲 Mx1；**H** 是一個 MxN 的通道矩陣，其中的每一個元素都是複數高斯分佈(Complex Gaussian)所形成的隨機變數；**d** 爲欲傳送的信號，矩陣大小爲 Nx1；**n** 代表的是一個 Mx1 的複數高斯雜訊。圖 16 可以用來說明整個解碼的過程。

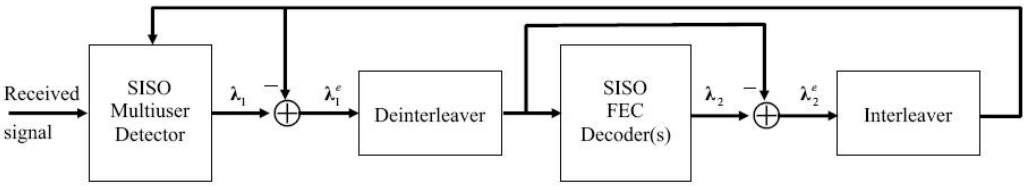


圖 16 渦輪式解碼器示意圖　[摘錄自 reference5]

圖 16 中 λ_1 、 λ_2 分別代表軟進-軟出多使用者偵測器(Soft-in Soft-out Multi-user detector)以及軟進-軟出前向式錯誤更正碼解碼器(Soft-in Soft-out Forward Error Correction Decoder)的軟資訊 (Soft Output)輸出，而 λ_1^e 、 λ_2^e 則代表了相對應的外部資訊(Extrinsic Information)。傳統上，軟進-軟出多使用者偵測器會先將軟式資訊(Soft Information)用來做軟式消除(Soft Canceling)，例如 V-BLAST 方法，並且在後面的解碼器使用最小平方誤差(Minimum Mean Square Error,MMSE)或是最大概似解(Maximum Likelihood solution,ML)解碼器。在此，我們並不會做軟式資訊的消除，而是將這些資訊送到下一級的軟進-軟出前向式錯誤更正碼解碼器中，利用

這些資訊來做解碼的動作。如圖 16 所示，多使用者偵測器與解碼器會不停地反覆交換軟式資訊，因此我們稱這個架構是渦輪式的(Turbo-Like)[5]。而解碼器可以使用摺積碼(Convolutional Code)或是渦輪碼(Turbo Code)。

爲了說明整個解碼的過程，我們先定義

$$\boldsymbol{\lambda}_1^e(k) = [\lambda_1^e(d_k(1))\lambda_1^e(d_k(2)).....\lambda_1^e(d_k(M))]^T \tag{5.2}$$

$$\boldsymbol{\lambda}_2^e(i) = [\lambda_2^e(d_1(i))\lambda_2^e(d_2(i)).....\lambda_2^e(d_K(i))]^T \tag{5.3}$$

其中 $\boldsymbol{\lambda}_1^e(k)$ 表示第 k 個使用者在某一個資料區塊(data block)中的外部資訊，而 $\boldsymbol{\lambda}_2^e(i)$ 代表第 k 個時間格(Time Slot)中所有使用者之外部資訊。而爲了說明方便，我們再假設傳送的資料符元(data symbol)只有+1 與-1 兩種可能。在這樣的條件下，每個符元的 LLR(Log Likelihood Ratio)可以式(5.4)、(5.5)表示。

$$\lambda_1(d_k(i)) = \log(\frac{P(d_k(i)=+1 \mid \mathbf{y}(i), \boldsymbol{\lambda}_2^e(i))}{P(d_k(i)=-1 \mid \mathbf{y}(i), \boldsymbol{\lambda}_2^e(i))}) \tag{5.4}$$

$$\lambda_2(d_k(i)) = \log(\frac{P(d_k(i)=+1 \mid \boldsymbol{\lambda}_1^e(k), decoding)}{P(d_k(i)=-1 \mid \boldsymbol{\lambda}_1^e(k), decoding)}) \tag{5.5}$$

(5.5)中的 LLR 值可以標準的渦輪解碼理論(Turbo Decoding Algorithm)來得解，因此我們的問題變成如何有效的求出式(5.4)中的 λ_1。爲了解釋求 λ_1 的困難度，我們將式(5.4)做一個簡單的推導：

$$\begin{aligned} P(d_k(i)=+1 \mid \mathbf{y}(i), \boldsymbol{\lambda}_2^e(i)) &= \sum_{\mathbf{d}_{-k}(i)} P(d_k(i)=+1, \mathbf{d}_{-k}(i) \mid \mathbf{y}(i), \boldsymbol{\lambda}_2^e(i)) \\ &= \sum_{\mathbf{d}_{-k}(i)} P(d_k(i)=+1 \mid \mathbf{y}(i), \boldsymbol{\lambda}_2^e(i), \mathbf{d}_{-k}(i)) P(\mathbf{d}_{-k}(i) \mid \mathbf{y}(i), \boldsymbol{\lambda}_2^e(i)) \end{aligned} \tag{5.6}$$

其中，第一個等式到第二個等式是利用機率的貝式定理，$\mathbf{d}_{-k}(i)$ 代表 $[d_1(i).....d_{i-1}(i), d_{i+1}(i), ...d_K(i)]^T$，而相加的對象包括所有 $d_k(i)=+1$ 的符元，當使用者越來越多時(即 K 越來越大)，其運算複雜度會是以指數型的方式增加，而想要直接求出式(5.6)也變得過於複雜以至於不能實現。在此我們利用馬可夫鏈-蒙地卡羅方法來解決這個問題，因這個方法在第二章已經詳細地介紹過了，在此我們只簡單地列出所用到的性質與結果。

由蒙地卡羅積分我們可以知道，欲求一函數 $h(x)$ 的期望值，可以利用其機率分佈 $f(x)$ 產生一些取樣點，再代入 $h(x)$ 即可，茲以下列式子表示：

$$E_f[h(x)] = \int_{\mathfrak{X}} h(x) f(x) dx \qquad f(x) \geq 0 \quad \forall x \in \mathfrak{X}$$

$$\Rightarrow E_f[h(x)] = \frac{1}{N_s} \sum_{n=1}^{N_s} h(x_n)$$

其中，$x_n \quad n = 1....N_s$ 爲 $f(x)$ 所產生的取樣點，而 N_s 爲取樣的個數。我們可以將(5.6)式以同樣地方法求出，如式(5.7)

$$P(d_k(i) = +1 \mid \mathbf{y}(i), \boldsymbol{\lambda}_2^e(i)) = \frac{1}{N_s} \sum_{n=1}^{N_s} P(d_k(i) = +1 \mid \mathbf{y}(i), \mathbf{d}_{-k}^n(i), \boldsymbol{\lambda}_2^e(i)) \tag{5.7}$$

其中，$\mathbf{d}_{-k}^n(i)$ 爲函數 $P(\mathbf{d}_{-k}(i) \mid \mathbf{y}(i), \boldsymbol{\lambda}_2^e(i))$ 的取樣點，也因此接下來我們會利用馬可夫鏈-蒙地卡羅方法中的吉布斯取樣(Gibbs Sampler)來對函數 $P(\mathbf{d}_{-k}(i) \mid \mathbf{y}(i), \boldsymbol{\lambda}_2^e(i))$ 做取樣。

1. 先隨機產生一組 $\mathbf{d}^{(0)}$,n=0
2. 接下來爲一個 for loop，可以產生取樣點 d_i 's
3. for n=1 to Ns

   由 $P(d_1 \mid d_2^{(n-1)}, d_3^{(n-1)}, ..., d_K^{(n-1)}, \mathbf{y}, \boldsymbol{\lambda}_2^e)$ 產生取樣點 $d_1^{(n)}$

   由 $P(d_2 \mid d_1^{(n)}, d_3^{(n-1)}, ..., d_K^{(n-1)}, \mathbf{y}, \boldsymbol{\lambda}_2^e)$ 產生取樣點 $d_2^{(n)}$

   .

   .

   .

   由 $P(d_K \mid d_1^{(n)}, d_2^{(n)}, ..., d_{K-1}^{(n)}, \mathbf{y}, \boldsymbol{\lambda}_2^e)$ 產生取樣點 $d_K^{(n)}$

吉布斯取樣會重複做 Ns 次，且經過了一段時間(這段時間稱爲訓練區間(Burn-in Period))的運行之後，所產生的取樣點 d_k 's 即爲機率分佈 $P(\mathbf{d}_{-k}(i) \mid \mathbf{y}(i), \boldsymbol{\lambda}_2^e(i))$ 的取樣點。

在求得 $\boldsymbol{\lambda}_1^e$ 後，我們不能將值直接代入第二級的前向式錯誤更正碼解碼器中，原因在於我們求出來的值是 LLR，而不是真正收到的訊號。然而，我們所接收的信號，本質上爲一個高斯隨機變數，其平均值爲我們所傳送的信號，而變異數則隨不同的通道雜訊而改變。而我們收到的數值則是這個機率分佈的一個取樣點，而我們可以將這個取樣點當作此機率分佈的平均值，原因是高斯分佈出現機率最高的值即爲此分佈之平均。有了這個概念之後，我們可以將 LLR 值轉換成這個位元的期望值，之後即可代入第二級的解碼器作運算。其過程如下：

$$L(b)=\log\left(\frac{p(b=+1|\mathbf{y})}{p(b=0|\mathbf{y})}\right)$$ ，b 即爲我們所要做轉換的位元。 (5.8)

$$p(b=+1|\mathbf{y})+p(b=0|\mathbf{y})=1 \quad (5.9)$$

由式(5.8)、(5.9)可推得

$$\Rightarrow \quad p(b=+1|\mathbf{y})=\frac{e^{L(b)}}{1+e^{L(b)}}$$

$$p(b=0|\mathbf{y})=\frac{1}{1+e^{L(b)}}$$

$$E[b]=1*p(b=+1|\mathbf{y})+(-1)*p(b=0|\mathbf{y})=\frac{e^{L(b)}}{1+e^{L(b)}}-\frac{1}{1+e^{L(b)}}=\tanh(L(b)/2) \quad (5.10)$$

其中

$$\tanh x=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}=\frac{e^{2x}-1}{e^{2x}+1}$$

5.2 使用表列式球狀解碼之渦輪式解碼器

球狀解碼也可以輸出軟式資訊，其系統如圖 17 所示。

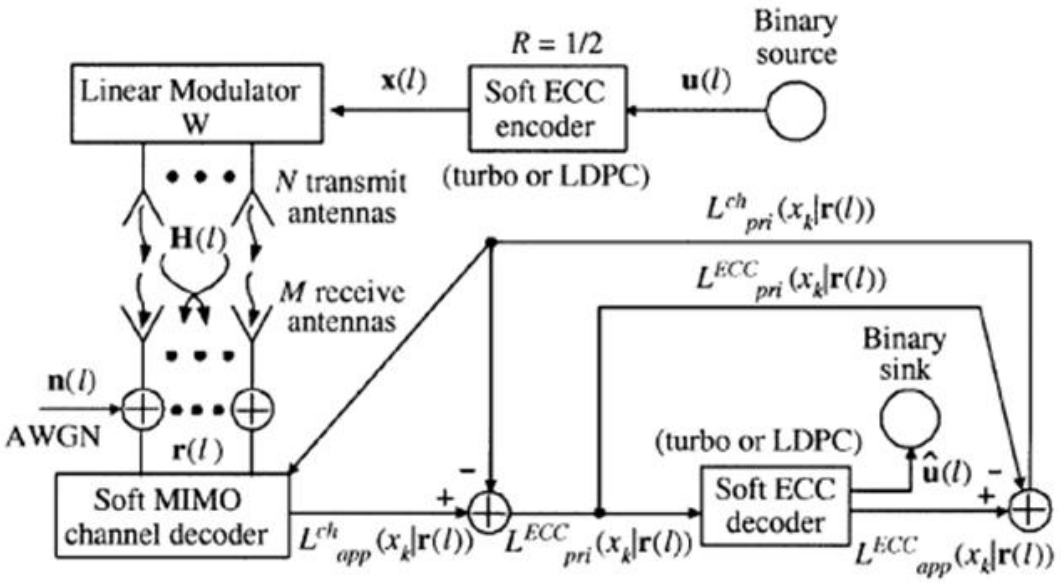


圖 17 軟式球狀解碼的系統架構圖[摘自 reference 9]

其中， $\mathbf{x}(l)$、$\mathbf{r}(l)$ 分別代表我們所傳送與接收的訊號， $L_{app}^{ch}(x_k \mid \mathbf{r}(l))$ 與 $L_{Ecc}^{pri}(x_k \mid \mathbf{r}(l))$ 分別對應到 5.1 節中所提到的 λ_1、λ_1^e；$L_{app}^{Ecc}(x_k \mid \mathbf{r}(l))$ 與 $L_{pri}^{ch}(x_k \mid \mathbf{r}(l))$ 則分別對應到 λ_2 、 λ_2^e 。此外， L_{pri}^{ch} 的物理意義爲所傳送的位元的事前機率，即在我們所使用的系統當中 0 與 1 的比例各占多少。在一般的情形下，我們都假設其爲等機率分佈； L_{app}^{ch} 的物理意義則是經過通道解碼之後，每個位元的後驗機率(A Posteriori Probability)，另外一個解釋方法爲提供通道的外部資訊給下一級的解碼器，也因此在式(5.8)中我們以 L_{ext}^{ch} 來代表他，而在圖中則是以 L_{pri}^{Ecc} 來表示，因爲他同樣可以看作是提供錯誤更正碼解碼器的事前機率。茲將上述的關係整理成式(5.8)、(5.9)、(5.10)：

$$\begin{aligned} L_{app}^{ch}(x_k \mid \mathbf{r}(l)) &:= \log\left(\frac{p(x_k = +1 \mid \mathbf{r}(l))}{p(x_k = -1 \mid \mathbf{r}(l))}\right) \\ &= L_{pri}^{ch}(x_k \mid \mathbf{r}(l)) + L_{ext}^{ch}(x_k \mid \mathbf{r}(l)) \end{aligned} \tag{5.11}$$

$$L_{pri}^{ch} = \log(\frac{p(x_k = +1)}{p(x_k = -1)}) \tag{5.12}$$

$$\begin{aligned} L_{pri}^{Ecc} = L_{ext}^{ch}(x_k \mid \mathbf{r}(\ell) &= \log(\frac{\sum_{x_k=+1} p(\mathbf{r}(l) \mid \mathbf{x}) \exp(\frac{1}{2}\mathbf{x}_{[k]}^T L_{pri[k]}^{ch})}{\sum_{x_k=-1} p(\mathbf{r}(l) \mid \mathbf{x}) \exp(\frac{1}{2}\mathbf{x}_{[k]}^T L_{pri[k]}^{ch})}) \\ &\cong \max_{x_k=+1}\{-\frac{1}{2\sigma^2}\left\|Q^H\mathbf{r} - R\mathbf{x}\right\| + \frac{1}{2}\mathbf{x}_{[k]}^T L_{pri[k]}^{ch}\} - \\ &\quad \max_{x_k=-1}\{-\frac{1}{2\sigma^2}\left\|Q^H\mathbf{r} - R\mathbf{x}\right\| + \frac{1}{2}\mathbf{x}_{[k]}^T L_{pri[k]}^{ch}\} \end{aligned} \tag{5.13}$$

其中

$$\mathbf{x}_{[k]} = [x_1, x_2, ...x_k, ...x_n]$$

$$p(\mathbf{r}(l) \mid \mathbf{x}) = \frac{\exp^{-\frac{1}{2\sigma^2}\left\|Q^H\mathbf{r} - R\mathbf{x}\right\|}}{(2\pi\sigma^2)^M} \tag{5.14}$$

由式(5.13)我們可以知道，當我們在做表列式球狀解碼時，不單單只是將可能的解儲存，而更要將其每一項的$\left\|Q^H\mathbf{r} - R\mathbf{x}\right\|$儲存起來。因爲對球狀解碼器而言，$\left\|Q^H\mathbf{r} - R\mathbf{x}\right\|$就是他所得出信號的軟式資訊。原因如式(5.12)所示，我們所接受到的訊號$\mathbf{r}(l)$，其機率分佈是呈現一個彼此互相獨立且相同(Independent and Identical, *i.i.d*)的多維高斯分佈。當然同樣地，我們在將軟式資訊送入第二級的解碼器時，仍然需要做雙曲正切(Hyper-Tangent)的轉換。

第六章　模擬結果

6.1 模擬環境與參數

Modulation	BPSK,QPSK,16QAM,64QAM
Number of Transmitted Antenna	4,8
Number of received Antenna	4
Number for Gibbs Sampler	50
Number for Burn-in Period	20
Encoding Scheme	(5,7)Convolutional Code
Channel	Fast Rayleigh Fading
Interleaver	S-random 1024

6.2 模擬結果與討論

6.2.1 無編碼之傳送效能

由圖 18 可以看出在沒有錯誤更正碼的幫忙下，馬可夫鏈-蒙地卡羅方法雖然仍能降低錯誤率，但其下降的幅度並不能讓人滿意，也因此並沒有看到有相關的論文是單獨地以馬可夫鏈-蒙地卡羅方法來進行解調。其原因在於馬可夫鏈-蒙地卡羅方法並沒有錯誤更正的能力。當我們所收到的信號因雜訊干擾，而使得正確的取樣點出現的機率很低，則在有限的取樣點數中，很有可能產生的皆爲錯誤的取樣點，而這也解釋了爲何軟式維特比解碼可以改善其錯誤率。軟式維特比解碼器可以提供此位元的信賴程度，也使得在產生取樣點時的正確率提高很多。

而由表 2 可以看出在 BPSK 的系統架構下，馬可夫鏈-蒙地卡羅的複雜度與球狀解碼的複雜度並沒有相差很多。原因在於馬可夫鏈-蒙地卡羅方法需要一段訓練時間(Training Period)，約 50 個取樣點。在這樣的情形下，由於我們的模擬是以 BPSK 來進行，因此就運算複雜度上不會相差很多。然而，當以更複雜的系統，例如 QPSK、16QAM 等，馬可夫鏈-蒙地卡羅方法的複雜度只會以線性上升，相較於其他幾種接收端解調方法，其複雜度就會有明顯地差別。

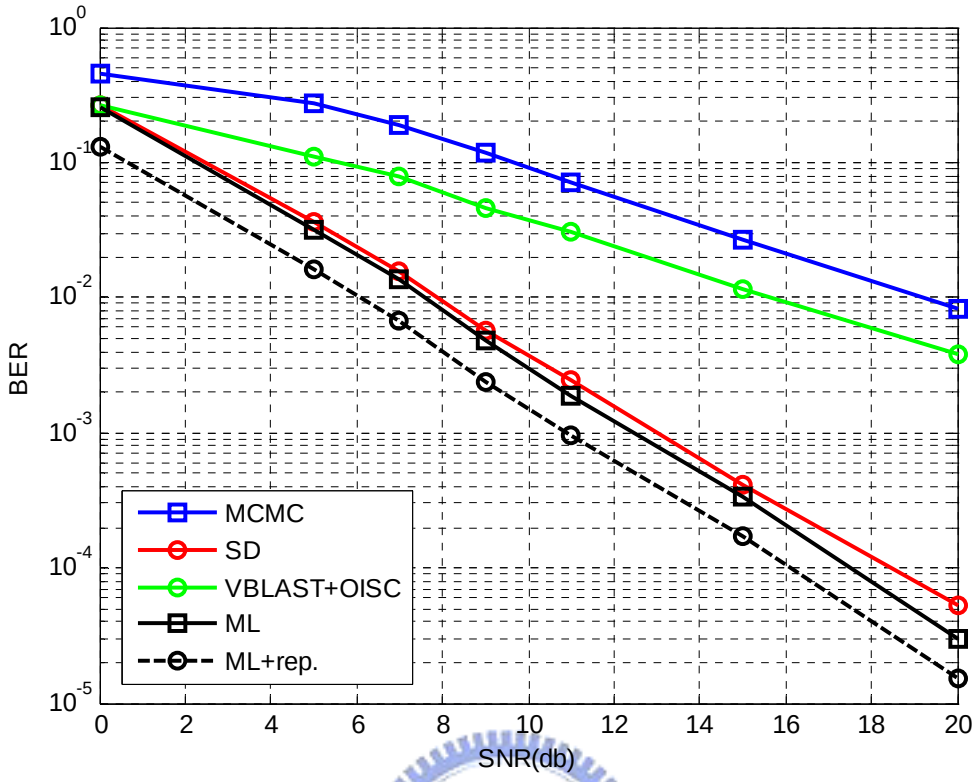


圖 18 沒有錯誤更正碼的馬可夫鏈-蒙地卡羅方法與其他系統的比較

	ADD	Multiply
ML	2000000	2400000
Sphere	1051398	2720263
Gibb	1400000	1600000
VBLAST	2200000	2200014

表格 2 傳送10^4個位元的運算複雜度比較表

此外，要補充說明的一點是，球狀解碼需要用到 Matlab 中內建的函數 QR 分解，而由 Matlab 的說明手冊中可以知道，其 QR 分解是採用 lapack 方法來進行運算，而每做一次運算其加法與乘法的複雜度分別爲

$$\text{乘法}: mn^2 - \frac{1}{3}n^3 + mn + \frac{1}{2}n^2 + \frac{23}{6}n$$

$$\text{加法}: mn^2 - \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{5}{6}n$$

假設運算的對象是一個 m 列 n 行的矩陣。

6.2.2 有編碼之傳送效能

圖 19 完全依照論文[5]的步驟來執行，其目的在於驗證對於渦輪式解碼的了解度以及系統程式的正確性。此外，由圖中我們可以發現在迴圈執行次數少的時候，位元錯誤率(Bit Error Rate)非常的高，原因在於我們的傳送天線比接收天線來得多，因此所形成的聯立方程是不足以將原始的資料解出來。然而，隨著迴圈執行次數增加，訊號中所含有的軟資訊可以一再地被利用，而能幫助我們將原本無法解出的信號解出。由論文的描述中我們可以知道，即使是 MMSE 解調，在八根傳送天線，四根接收天線的系統之下，他的位元錯誤率依然不佳，而藉由渦輪式解碼在位元錯誤率是10^{-3}時可以得到約 2db 的增益。

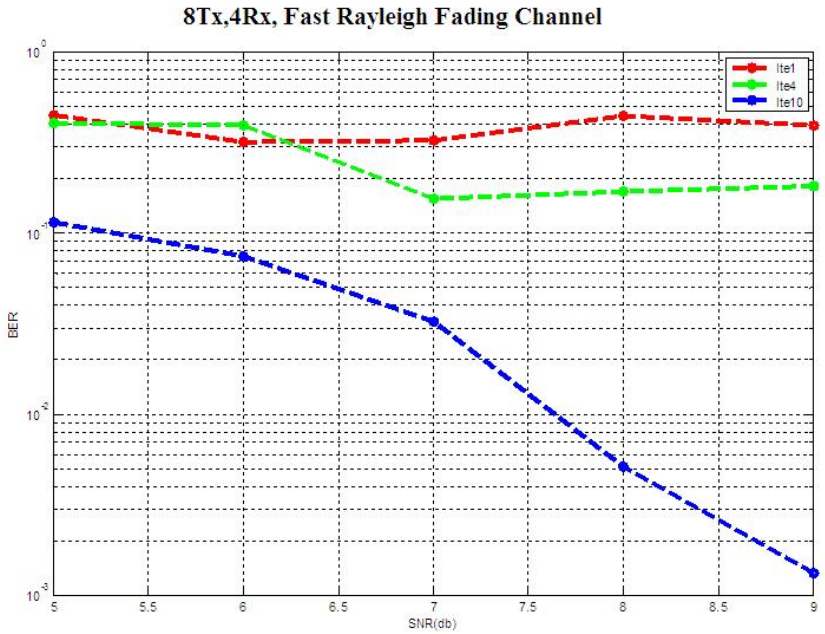


圖 19　8 根傳送天線，4 根接收天線之效能圖

圖 20 說明了在 4 根傳送天線，4 根接收天線的系統下，還沒有進入迴圈(即 iteration=1)的渦輪式解碼器即具有較單純的 ML 解更佳的位元錯誤率表現。而當迴圈數增加，錯誤率的改善也是非常地明顯。

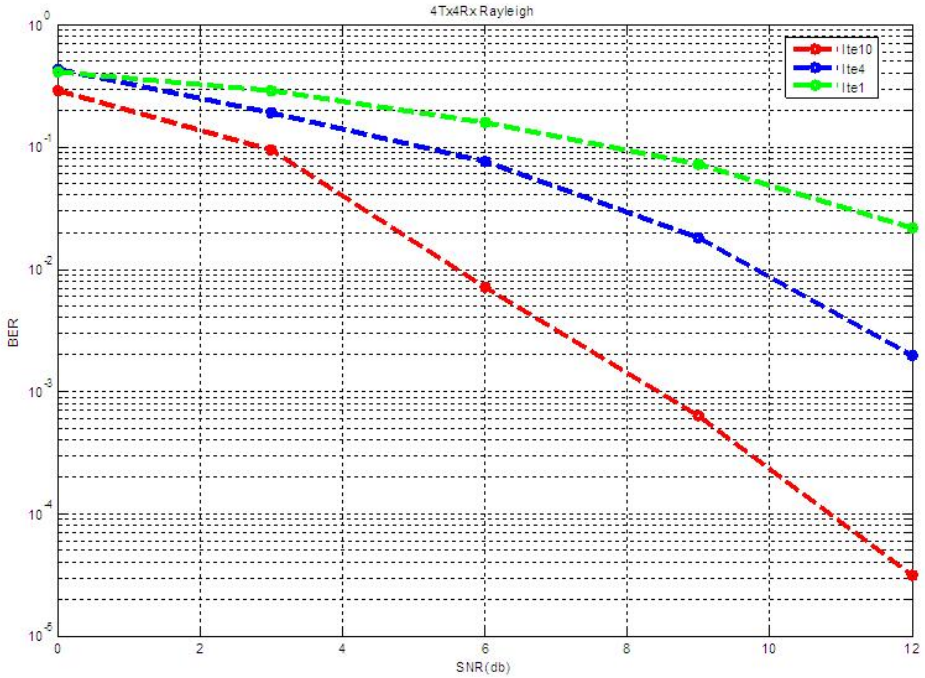


圖 20 4 根傳送天線，4 根接收天線的系統效能圖

圖 21 比較兩種不同的多使用者偵測器的錯誤率表現，包括馬可夫鏈-蒙地卡羅(MCMC)以及表列式球狀解碼(LSD)兩種方法。由 ML 與 LSD 的曲線我們可以看出，在沒有編碼的情況之下，LSD 的表現是跟 ML 相當的。而在渦輪式解碼的架構下，MCMC 與 LSD 都可以達到不錯的效能。同時根據其他的論文的模擬結果，在 BPSK 與 QPSK 的情形之下，LSD 與 MCMC 的錯誤率曲線應是接近重合的，這個模擬結果也許其他的文獻記載相符。而由表格 3 我們可以知道，兩種方法的運算複雜度均較 ML 少了一半。

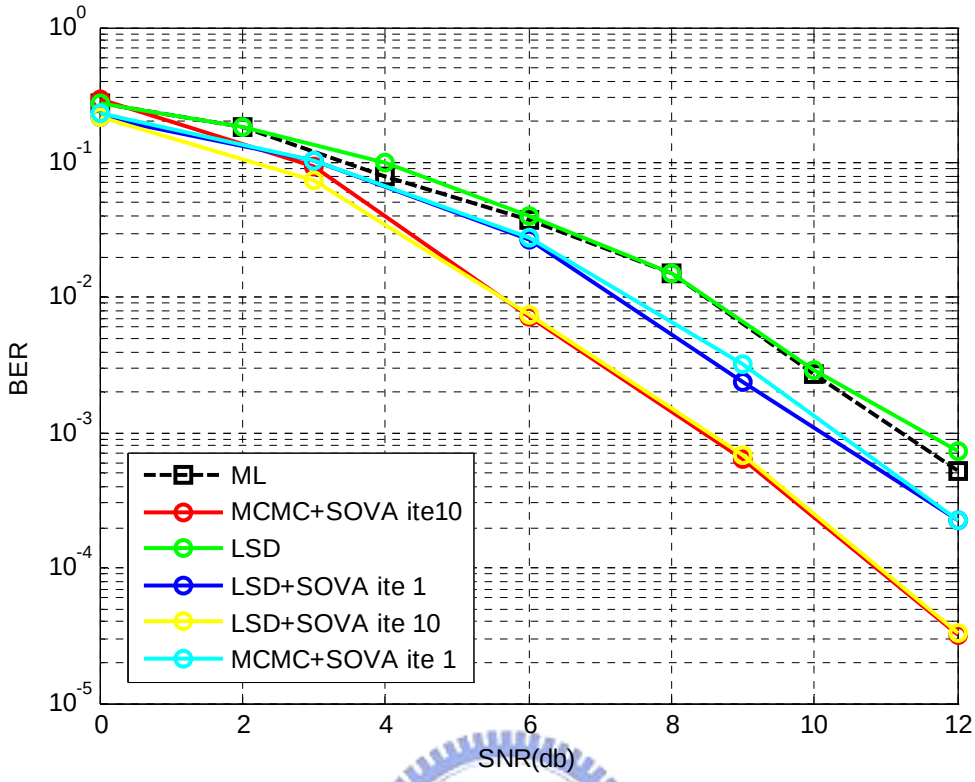


圖 21 4 傳 4 收系統下，各種方式的錯誤率比較(BPSK)

	ADD	Multiply
ML	$9.12*10^6$	$7.31*10^6$
MCMC+SOVA Iteration 10	$3.12*10^6$	$4.7*10^6$
LSD+SOVA Iteration 10	$5.71*10^6$	$6.32*10^6$

表格 3 傳送10240個位元的運算複雜度比較表(BPSK)

圖 22,23 以及表格 4,5 與圖 21 所用的模擬環境一樣，差別在於調變端改爲使用高維度的 16QAM 及 64QAM 調變以提升資料傳輸的效率。由這個模擬可以觀察得知，在高維度時 MCMC 方法運算複雜度的優勢會明顯地展現出來。

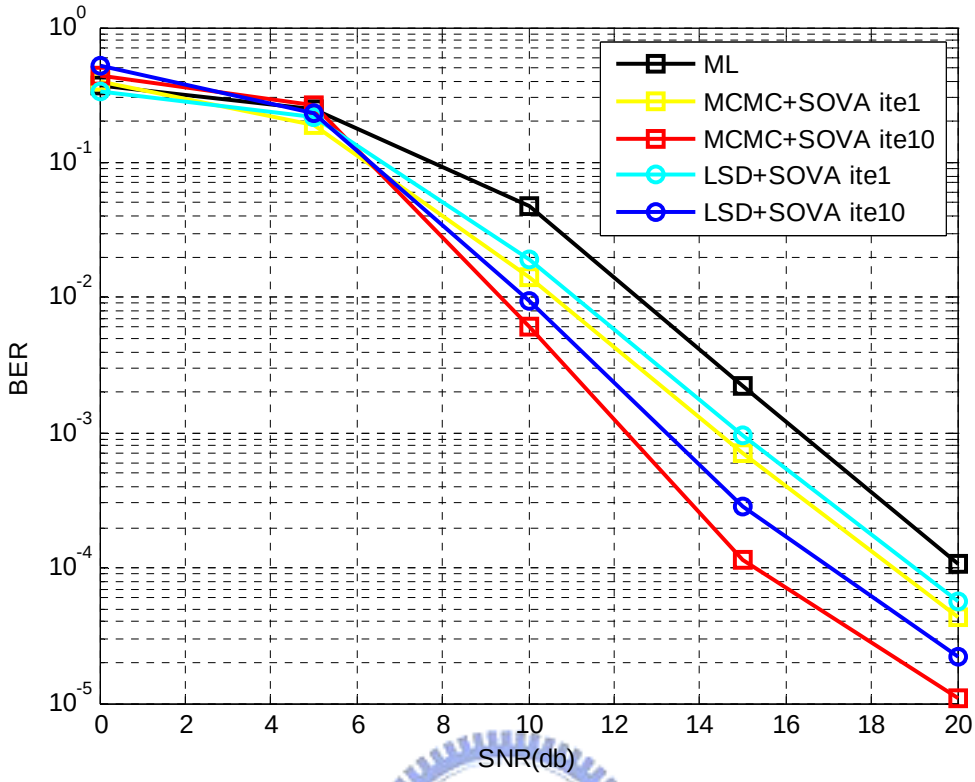


圖 22 4 傳 4 收系統下，各種方式的錯誤率比較(16QAM)

	ADD	Multiply
ML	$6.41*10^8$	$5.89*10^8$
MCMC+SOVA Iteration 10	$2.83*10^7$	$1.47*10^8$
LSD+SOVA Iteration 10	$6.81*10^8$	$5.15*10^8$

表格 4 傳送10240個位元的運算複雜度比較表(16QAM)

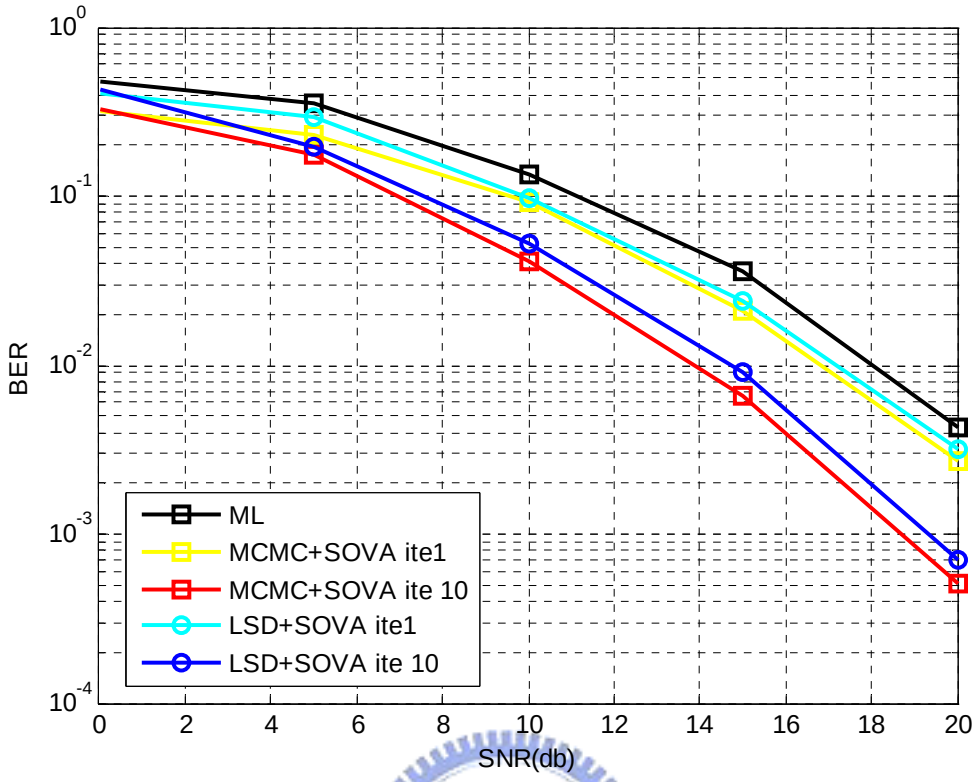


圖 23 4 傳 4 收系統下，各種方式的錯誤率比較(64QAM)

	ADD	Multiply
ML	$7.81*10^{10}$	$9.17*10^{10}$
MCMC+SOVA Iteration 10	$8.12*10^{8}$	$8.71*10^{8}$
LSD+SOVA Iteration 10	$4.9*10^{10}$	$7.12*10^{10}$

表格 5 傳送10240個位元的運算複雜度比較表(64QAM)

第七章　結論與未來方向

在這篇論文中，我們探討了一些馬可夫鏈-蒙地卡羅方法的性質，並將其應用在多輸入多輸出的系統上。藉由軟式解碼器的協助，我們可以簡單的架構來解調整個複雜的系統。而與表列式球狀解碼相比，效能也相差無幾，但式運算複雜度卻得到大量的減低。自 2002 年以降，已經有非常多的論文在討論馬可夫鏈-蒙地卡羅方法應用在實際解調系統上的效能與方式，同時也有看到如何設計相關電路圖來與之配合。在 802.11n 的標準當中，原創者也有將此方法提出，並獲得接受。在可預見的未來，相信仍然是有非常多的發展的。

參考文獻

[1] Richard van Nee , ”***OFDM for Wireless Communications***”, Artech House,2000

[2] George Casella and Edward I. George, “***Explaining the Gibbs Sampler***”,The *American Statistician*, Vol. 46, No. 3, (Aug., 1992), pp. 167-174

[3] Siddhartha Chib and Edward Greenberg ,“***Understanding the Metropolis-Hastings Algorithm***”, *The American Statistician,* Vol. 49, No. 4, (Nov., 1995), pp. 327-335

[4] Gelfand, A.E. and Smith, A.F.M. (1990). “***Sampling-based approaches to calculating marginal densities***”. *Journal of the American Statistical Association*, 85: 398-409.

[5] Zhenning Shi, Haidong Zhu, Behrouz Farhang-Boroujeny,” ***Markov Chain Monte Carlo Techniques in Iterative Detectors: A Novel Approach Based on Monte Carlo Integration***”, Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE,Volume 1, 29 Nov.-3 Dec. 2004 Page(s):325 - 329 Vol.1

[6] Tuchler, M.; Koetter, R.; Singer, A.C.,”***Turbo equalization: principles and new results***” *Communications*, *IEEE Tran., Volume 50, Issue 5*, May 2002 Page(s):754 - 767 Digital Object Identifier 10.1109/TCOMM.2002.1006557

[7] R. Koetter, A.C. Singer and M. Tuechler, “***Turbo Equalization***”,*Signal Processing Magazine*, invited paper, 2003

[8] Branka Vucetic, and Jinhong Yuan, “ T***urbo Codes –Principles and Applications***“,Kluwer Academic Publisher, 2000

[9] Baldur Steingrimsson, Zhi-Quan Luo, Kon Max Wong, “***Soft Quasi-Likelihood Detection for Mulriple-Antenna Wireless Channels***”, IEEE Transaction On Signal Processing,Vol. 51, No. 11, Nov. 2003.

[10] L. G. Barbero and J. S. Thompson, "***Performance Analysis of a Fixed-Complexity Sphere Decoder in High-Dimensional MIMO Systems***", in IEEE International Conference on Acoustics, *Speech, and Signal Processing* (ICASSP '06), vol. 4, Toulouse, France, May 2006, p. 557-560

[11] Andrew J. Viterbi,”***Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”***, IEEE Transactions on Information Theory 13(2):260–269, April 1967.

[12] L.Hanzo,T.H.Liew,B.L.Yeap,”***Turbo Coding,Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels”,***Wiley.

[13] J. Hagenauer, P. Hoeher,”***A Viterbi algorithm with soft-decision outputs and its applications”***, Proc. IEEE GLOBECOM, pp. 47.11-47.17, Dallas, TX, Nov

1989.
[14] Cowles and Carlin, "***Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review***",1996
[15] Babak Hassibi,Haris Vikalo, "***On the Sphere Decoding Algorithm I. Expected Complexity***", IEEE Transactions on Signal Processing,Vol. 53, No. 8,Aug. ,2005
[16] Sylvain Ranvier,"***Overview of MIMO systems***", course note.
[17] Bertrand M. Hochwald Stephan Ten Brink," ***Achieving Near-Capacity on a Multiple-Antenna Channel***", Dec 9, 2002.
[18] Andrew J. Viteerbi,"***An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes***", IEEE journal on selected area in communications, Vol.16, No.2, Feb. 1998
[19] G. Forney,"***The Viterbi Algorithm***",Proceedings of the IEEE,,Vol 61,pp268-278,March 1973