國 立 交 通 大 學

電信工程學系

碩 士 論 文

里德所羅門碼之信念傳遞解碼演算法研究

A Study on Belief-Propagation Based Decoding Algorithms for Reed-Solomon Codes

研 究 生：謝郁民

指導教授：王忠炫

中 華 民 國 九十六 年 七 月

里德所羅門碼之信念傳遞解碼演算法研究

A Study on Belief-Propagation Based Decoding Algorithms for Reed-Solomon Codes

研 究 生：謝郁民　　Student：Yu-Min Hsieh

指導教授：王忠炫　　Advisor：Chung-Hsuan Wang

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

Master of Scince

in Communication Engineering

July 2007

Hsinchu, Taiwan, Republic of China

中 華 民 國 九十六 年 七 月

里德所羅門碼之信念傳遞解碼演算法研究

研究生：謝郁民　　指導教授：王忠炫 博士

國立交通大學

電信工程學系碩士班

摘　　要

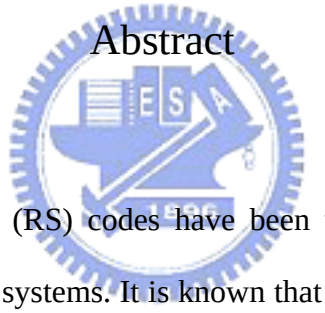
里德所羅門碼 (Reed-Solomon codes) 目前已被使用於許多先進的通訊和儲存系統上。眾所皆知，里德所羅門碼的軟式決策解碼相較於傳統的硬式決策解碼能夠提供明顯的效能改善，但由於其軟式決策解碼的高複雜度，目前大部分的系統仍然使用傳統的硬式決策解碼器。Jiang 和 Narayanan 於 2006 年提出了一種利用適應性校驗矩陣，對里德所羅門碼進行軟式輸出輸入解碼的演算法 (JN 演算法)。JN 演算法能夠避免校驗節點飽和化以及低信任度位置之錯誤傳遞等問題，然而其潛在的問題在於過度信任高信任度位置的訊息。於本篇論文中，我們首先將討論造成 JN 演算法解碼失敗的因素，並提出適當的演算法來避免 JN 演算法中高信任度位置錯誤所造成的影響，並增進其解碼效能。

A Study on Belief-Propagation Based Decoding Algorithms for Reed-Solomon Codes

Student：Yu-Min Hsieh　　Advisor：Dr. Chung-Hsuan Wang

Department of Communication Engineering

National Chiao Tung University



Abstract

Recently, Reed-Solomon (RS) codes have been used in many state-of-the art communication and recording systems. It is known that soft-decision decoding (SDD) of RS codes provides significant performance gain over hard decision decoding (HDD), but most systems still are based on HDD because of the high complexity of SDD. In 2006, Jiang and Narayanan presented an iterative soft-in soft-out decoding algorithm of RS codes by adapting the parity check matrix. JN algorithm can avoid the problem of check node saturation and the error propagation from the least reliable positions, but the drawback of JN algorithm is to over believe the message from the most reliable positions. In this thesis, we first discuss the cause of decoding failure in traditional JN algorithm. Based on our discussion, we present a proposed algorithm to avoid the influence of the high reliable errors in JN algorithm and improve the decoding performance.

誌　謝

本篇論文得以完成，首先要感謝指導教授王忠炫博士的栽培，在兩年的過程中，教導我研究的態度和方法，並且適時地給予建議和方向。也感謝同學、學弟、學長們的鼓勵和幫助。

特別想感謝在最後這段時間裡支持我的朋友們，小毛、軒哥、阿傑、冠霖，還有電工系的毛大學長，蘭友會的朋友們，在精神上給予極大的支持和幫助。最後想謝謝我的家人，讓我能夠專心在研究過程中，謝謝大家。

民國九十六年七月

研究生謝郁民謹識於交通大學

目錄

圖目錄

第一章

概論

里德所羅門碼 (Reed-Solomon codes) [1] 為最大距離分離碼 (maximum distance separable code，簡稱 MDS 碼)，目前已被大量應用於數位通訊系統和儲存系統上。對一個(N,K)里德所羅門碼而言，使用傳統的硬式決策 (hard-decision) 解碼器能夠更正碼字中個數為$t=\lfloor d_{\min}/2 \rfloor$以內的錯誤，其中$d_{\min}=N-K+1$為$(N,K)$里德所羅門碼的最小距離 (minimum distance)。由於里德所羅門碼本身的碼字為非二位元符元 (non-binary symbol) 所組成，因此有可能在更正一個符元的同時，更正了數個連續的錯誤位元，故其極適合用於處理突發錯誤 (burst errors) 的情況。

大部分的使用上，由於高速和低複雜度的需求，里德所羅門碼的解碼方式多採用硬式決策解碼器，其利用代數結構的特性得到一組解碼方程式，並且使用有限場 (finite field) 的數學運算來解出方程式的未知數。然而眾所皆知，里德所羅門碼的最大概似解碼器 (maximum likelihood decoder) 相較於傳統的硬式解碼器能夠提供顯著的效能改善，但最大概似解碼器的高複雜度問題，仍然沒有一個較為有效的處理方式。

面對這樣的問題，於軟式決策解碼方面 (soft-decision decoding)，Forney [2] 和 Chase [3] 曾提出基於信任度的解碼方式，利用翻轉碼字中的最小信任位置 (least reliable positions，簡稱 LRPs) ，試圖尋找和接收信號相關性最大的合法碼字。同樣地，利用信任度解碼的方式，Fossorier 和 Lin [4] 利用尋找最大獨立信任位置 (most reliable independent positions，簡稱 MRIPs)，並重新編碼以尋找和

接收信號相關性最大的碼字，這種方法稱為排序統計解碼演算法 (ordered statistic decoding algorithm，簡稱 OSD)。Vardy 和 Ber'ey [5] 利用里德所羅門碼代數上的特性，將其生成矩陣 (generator matrix) 經二位元映射後，拆解為數個 BCH 碼為其子場域子碼 (subfield subcode)，並以連接碼 (glue code) 作為其之間的連繫。僅管這種方式降低了使用柵欄解碼 (trellis decoding) 的複雜度，然而整體複雜度還是會隨著碼長和 $d_{\min}$ 變大時快速增加。

而於疊代解碼 (iterative decoding) 方面，Ponnampalam 和 Grant [6] 利用和 Vardy 和 Ber'ey 相同的原理，將里德所羅門碼拆解為數個子柵欄 (sub-trellis)，並且利用之間連接碼的關係互相傳送消息量而進行疊代解碼。Liu 和 Lin [7] 則同樣是利用子場域子碼的拆解原理，將生成矩陣拆為數個大小較小的生成矩陣，並利用之間連接碼的關連性進行渦輪 (turbo) 編碼和解碼。同樣為疊代解碼的信念傳遞 (belief propagation) 演算法於低密度奇偶校驗碼 (LDPC codes) 提出後，廣泛受到討論。自然於里德所羅門碼的軟式決策解碼上，亦有相關的演算法被提出。2004 年 Jiang 和 Narayann 就提出了隨機平移解碼演算法 (stochastic shifting based iterative decoding scheme，簡稱 SSID) [8]。其利用里德所羅門碼經二位元映射後的二位元奇偶校驗矩陣 (binary parity check matrix) 和其為循環碼的特性，在每次疊代進行信念傳遞演算法中的梯度下降演算法 (gradient descent algorithm) 時，亦進行二位元奇偶校驗矩陣的隨機平移動作，以減少錯誤傳遞所造成的影響。Halford [9] 於此提出了延伸方向，其利用自同構 (automorphism) 的特性來增加冗餘二位元奇偶校驗矩陣 (redundant binary parity check matrix) 的個數，於每次疊代中隨機選擇一自同構冗餘奇偶校驗矩陣來進行梯度下降演算法。Hehn、Huber、Laendner 和 Milenkovic [10] 則是利用循環群產生器 (cyclic group generator) 生成數個二位元奇偶校驗矩陣後，平行進行信念傳遞演算法，從中選擇和接收信號相關性最大的碼字輸出。

2004 年 Jiang 和 Narayan [11] 提出了 JN 演算法，其方式為對二位元奇偶校驗矩陣依照信任度進行高斯消去動作，而減少未知位置 (erasure positions) 對於

梯度下降演算法所造成的影響。而 Kothiyal、Takeshita 和 Fossorier [12] 提出了結合 OSD(1) 演算法和適應性信念傳遞 (adaptive belief propagation，簡稱 ABP) 的機制來相互輔助，稱為 ABP-OSD(1)演算法，其 ABP 的動作可為 JN 演算法或 [13]。

本研究主要針對 JN 演算法進行討論，利用修正 ABP-OSD(1)的機制，以解決 JN 演算法的潛在問題，並利用數據觀察說明提出之演算法對於原有的 JN 解碼演算法所帶來的效應，最後呈現經由提出之演算法所帶來的效能增益和複雜度考量。本論文之架構如下：第二章中將會介紹相關的符號定義、里德所羅門碼的二位元映射方法，以及梯度下降演算法之理論基礎。第三章中將簡介 JN 解碼演算法的解碼步驟和相關的改善方式，並且介紹 ABP-OSD(1) 的解碼機制和運作方式。第四章中，一開始將討論 JN 解碼演算法的潛在問題，並且針對其問題來說明本研究所提出之演算法，最後佐以數據觀察經提出之演算法作用後，對於原有 JN 解碼演算法之影響。第五章將呈現提出之演算法的效能表現並進行複雜度討論。最後於第六章中提出本論文之結論。

第二章

梯度下降解碼演算法之理論基礎

於本章中將定義相關的符號標示法和所處之系統環境，並介紹里德所羅門碼之二位元映射法，接下來將說明梯度下降解碼演算法的理論基礎，此理論基礎將使用於接下來的章節，最後以圖形化說明其運作方式。接下的說明中，於字母下方加一底線代表一個向量，如 $\underline{c}$；粗體的字母代表一個矩陣，如 $\mathbf{H}$。向量或矩陣中的元素以下標來表示，如 c_i 表示向量 $\underline{c}$ 中的第 i 個元素，H_{ij} 表示矩陣 $\mathbf{H}$ 中第 i 列第 j 行的元素。

2-1 符號定義

2.1.1 傳送信號符號定義

首先定義一個二位元碼字 (binary codeword) $\underline{c}=[c_1,c_2,\ldots,c_n]$，碼字長度為 n，碼字空間的維度 (dimension) 以 k 來表示。向量中每個元素 c_i 均屬於 $GF(2)$ 中的元素。本研究以 $\mathbf{H}$ 來表示這個碼字空間 (codeword space) 的奇偶校驗矩陣 (parity check matrix)，並以 μ 代表 $\mathbf{H}$ 的列數，n 為 H 的行數；$\underline{h}_i$ 則表示矩陣中第 i 個奇偶校驗方程式 (parity check equation)，亦為 $\mathbf{H}$ 的第 i 列向量。

一個合法的碼字必須要滿足 $\mathbf{H}$ 中每一個奇偶校驗方程式偶校驗 (even parity check) 的特性。所謂的偶校驗，是指當選取奇偶校驗方程式 $\underline{h}_i$，並從 $\underline{h}_i$ 中為 1 的位置取出對應位置的 c_i，則這些 c_i 的值在 $GF(2)$ 的加法下，必須總和為 0。如

下列之說明：

$\mathbf{H}=\begin{bmatrix}1&1&0&1&1&0&0\\1&1&1&0&0&1&0\\1&0&1&1&0&0&1\end{bmatrix}$，而 $\underline{c}=[1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0]$，符合所有的 $\mathbf{H}$ 中的

奇偶校驗方程式 $\begin{cases}1+1+\ \ +1+1+\ \ +\ \ =0\\1+1+0+\ \ +\ \ +0+\ \ =0\\1+\ \ +0+1+\ \ +\ \ +0=0\end{cases}$，故 $\underline{c}$ 為一合法碼字。

2-1.2 通道環境符號定義

在本論文中假設所使用的通道環境均為可加性高斯白雜訊 (additive white Gaussian noise) 通道，以下均簡稱為 AWGN 通道；調變方式為二位元移鍵 (binary phase shift keying) 調變，以下均簡稱為 BPSK。BPSK 調變將 0 映射至+1 且 1 映射至-1 後，最後傳送於 AWGN 通道的過程可以由 (2.1) 式所表示：

$$\underline{y}=(-2\underline{c}+1)+\underline{n}. \tag{2.1}$$

其中 $\underline{y}=[y_1,y_2,.....,y_n]$ 表示經過 BPSK 調變和 AWGN 通道後的接收信號，而 $\underline{n}=[n_1,n_2,.....,n_n]$ 表示 AWGN 通道上的雜訊，且其雙邊雜訊功率為 $N_0/2$。

獲得接收信號後，即可計算碼字中每個元素的對數概似比例 (log likelihood ratio，以下均簡稱為 LLR)，計算的方式為 (2.2) 式：

$$L(c_i)=\log\frac{\Pr(c_i=0\mid y_i)}{\Pr(c_i=1\mid y_i)}. \tag{2.2}$$

於 AWGN 通道，可將 (2.2) 式化簡為 (2.3) 式：

$$L(c_i)=\log\frac{\Pr(c_i=0\mid y_i)}{\Pr(c_i=1\mid y_i)}=\log\frac{\frac{1}{\sqrt{\pi N_0}}e^{\frac{(y_i-1)^2}{N_0}}}{\frac{1}{\sqrt{\pi N_0}}e^{\frac{(y_i+1)^2}{N_0}}}=\frac{4y_i}{N_0}. \tag{2.3}$$

在接下來的討論中，演算法所使用的初始 LLR，均以 (2.3) 式計算之。

2-1.3 里德所羅門碼之符號定義

(N,K) 里德所羅門碼運作於 $GF(2^m)$ 之下時，則長度為 $N=2^m-1$，維度為 K，則可求得其最小距離為 $d_{\min}=N-K+1$。而奇偶校驗矩陣可用 (2.4) 式的形式來表示：

$$\mathbf{H}_S=\begin{bmatrix} 1 & \beta & \cdots & \beta^{(N-1)} \\ 1 & \beta^2 & \cdots & \beta^{2(N-1)} \\ & & \cdots & \\ 1 & \beta^{(d_{\min}-1)} & \cdots & \beta^{(d_{\min}-1)(N-1)} \end{bmatrix}. \tag{2.4}$$

$\mathbf{H}_S$ 為 M 列 N 行之矩陣，在這裡 β 為 $GF(2^m)$ 中的質元素 (primitive element)。

2-2 里德所羅門碼之二位元映射 [14]

2-2.1 二位元映射基底

本研究中，以 $GF(2^m)$ 中的符元 (symbol) $1,\beta,\beta^2,\cdots,\beta^{m-1}$ 做為符元映射至二位元向量之基底。如 $A\in GF(2^m)$ 為一符元，其可投影在 $1,\beta,\beta^2,\cdots,\beta^{m-1}$ 所組成的基底上，而以 $A=\sum_{i=0}^{m-1}A_b^{(i)}\beta^i$ 或 $\underline{A}_{\text{vector}}=\begin{bmatrix}A_b^{(0)} & A_b^{(1)} & \cdots & A_b^{(m-1)}\end{bmatrix}$ 的形式來表示，其中 $A_b^{(i)}\in GF(2)$。

範例

於 $GF\left(2^3\right)$ 時，使用的質多項式 (primitive polynomial) 為 $1+x+x^3$，則 $GF\left(2^3\right)$ 中的每個元素均可用 $GF\left(2\right)[x]\bmod\left(1+x+x^3\right)$ 的方式表示如下，其中 $\beta=x$：

$$0 = 0+0\cdot x+0\cdot x^2 = [0,0,0],\ 1 = 1+0\cdot x+0\cdot x^2 = [1,0,0],$$
$$\beta = 0+1\cdot x+0\cdot x^2 = [0,1,0],\ \beta^2 = 0+0\cdot x+1\cdot x^2 = [0,0,1],$$
$$\beta^3 = 1+1\cdot x+0\cdot x^2 = [1,1,0],\ \beta^4 = 0+1\cdot x+1\cdot x^2 = [0,1,1],$$
$$\beta^5 = 1+1\cdot x+1\cdot x^2 = [1,1,1],\ \beta^6 = 1+0\cdot x+1\cdot x^2 = [1,0,1].$$

2-2.2 代數加法之二位元映射

每個符元映射為二位係數的向量後，仍然需要滿足在$GF(2^m)$中的加法特性。由於先前選取的基底 $1,\beta,\beta^2,\cdots,\beta^{m-1}$，其在加法上可直接由向量的$GF(2)$二位元加法取代原有的符元加法計算，因此經過二位元映射後，加法仍然滿足原有的代數特性。

範例

於$GF(2^3)$中

$$\beta+\beta^5 = (0+1\cdot x+0\cdot x^2)+(1+1\cdot x+1\cdot x^2) = [0,1,0]+[1,1,1]$$
$$= [1,0,1] = 1+0\cdot x+1\cdot x^2 = \beta^6.$$

2-2.3 代數乘法之二位元映射

符元映射為二位元向量後，亦需要滿足在$GF(2^m)$中的乘法特性。而在乘法上，每個符元經二位元映射後的二位元向量乘上另一符元的動作，在這裡需要以乘上一矩陣的動作來取代，以維持原有的代數特性。我們藉由原先選擇的基底則可如範例中的方式推得每個符元在乘法上所代表的矩陣。

範例

於$GF(2^3)$中

$$\beta^5\cdot 1 = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}[1,0,0]^T = [A,D,G]^T = \beta^5 = [1,1,1]^T.$$

$$\beta^5 \cdot \beta = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} [0,1,0]^T = [B,E,H]^T = \beta^6 = [1,0,1]^T .$$

$$\beta^5 \cdot \beta^2 = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} [0,0,1]^T = [C,F,I]^T = \beta^7 = 1 = [1,0,0]^T .$$

故我們可知 β^5 在經過二位元映射後，乘法上可用 $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$ 來取代。

而由以上的推論，可得到 $GF(2^3)$ 中所有符元所代表的乘法矩陣如下：

$$0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},\ 1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ \beta = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},\ \beta^2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix},$$

$$\beta^3 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix},\ \beta^4 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix},\ \beta^5 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix},\ \beta^6 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

2-2.4 里德所羅門碼二位元奇偶校驗矩陣之生成

$\mathbf{H}_s$ 可藉由 $GF(2^m)$ 中選擇的基底 $1, \beta, \beta^2, \cdots, \beta^{m-1}$，由原有的符元奇偶校驗矩陣映射成大小為 $(n-k)\times n$ 的二位元奇偶校驗矩陣 $\mathbf{H}_b$，在這裡 $n = m \cdot N$ 且 $k = m \cdot K$。經過二位元映射後，我們可以將里德所羅門碼的解碼問題，看待成大小為 $(n-k)\times n$ 的二位元線性區塊碼 (binary linear block code) 之解碼問題。

範例

於 $GF(2^3)$ 中，考慮一里德所羅門碼其 $N = 2^3 - 1 = 7,\ K = 5,\ d_{\min} = N - K + 1 = 3$，且奇偶校驗矩陣 $\mathbf{H}_S$ 為

$$\mathbf{H}_{\mathrm{S}}=\begin{bmatrix}1 & \beta & \beta^2 & \beta^3 & \beta^4 & \beta^5 & \beta^6\\ 1 & \beta^2 & \beta^4 & \beta^6 & \beta & \beta^3 & \beta^5\end{bmatrix}.$$

經過乘法映射後，則可以得到對應的二位元奇偶校驗矩陣 $\mathbf{H}_{\mathrm{b}}$，其 $n=21$ 且 $k=15$。

$$\mathbf{H}_{\mathrm{b}}=\begin{bmatrix}
\mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 1 & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} & 1 & 1 & 1 & \mathbf{1} & \mathbf{1} & \mathbf{0}\\
\mathbf{0} & \mathbf{1} & \mathbf{0} & 1 & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} & 1 & 1 & 1 & \mathbf{1} & \mathbf{1} & \mathbf{0} & 1 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1}\\
\mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 1 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{1} & 1 & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} & 1 & 1 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0}\\
1 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 1 & 1 & \mathbf{1} & \mathbf{1} & \mathbf{0} & 0 & 0 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} & 1 & 1 & 1\\
0 & 1 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{1} & 1 & 1 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & 1 & 0 & 1 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 1 & 0 & 0\\
0 & 0 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} & 1 & 1 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 1 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{1} & 1 & 1 & 0
\end{bmatrix}.$$

2-3 梯度下降法之理論基礎

以下將介紹 SSID 演算法和 JN 解碼演算法等演算法所使用的梯度下降法，其理論基礎 [12] 和執行步驟。在進行梯度下降法前，我們可以由式 (2.3) 得到接收信號中每個位置的 LLR ($L(c_i)$)，並且預備二位元奇偶校驗矩陣 $\mathbf{H}_{\mathrm{b}}$，以進行梯度下降法之解碼。

2-3.1 函數定義

接下來定義兩個函數以說明梯度下降演算法

函數定義 2-1

函數 $\nu:[-\infty,+\infty]\to[-1,+1]$ 為將 $L(c_i)$ 自 LLR 域映射至 tanh 域之函數：

$$\nu(L)=\tanh\left(\frac{L}{2}\right)=\frac{e^L-1}{e^L+1}. \tag{2.5}$$

此為一對一 (one-to-one) 且映成 (onto) 函數。

函數定義 2-2

ν 之反函數 $\nu^{-1}:[-1,+1]\to[-\infty,+\infty]$:

$$\nu^{-1}(t)=\ln\left(\frac{1+t}{1-t}\right)\ ,\ t\in[-1,+1]. \tag{2.6}$$

此亦為一對一且映成函數。

2-3.2 梯度下降法理論基礎

從接收信號求得每個位置的 LLR($L(c_i)$)後，首先將其利用式 (2.5) 將每個位置的 LLR 轉換至 tanh 域表示，如式 (2.7)：

$$\underline{T}=[T_1,T_2,\cdots,T_n]=\left[\nu\left(L(c_1)\right),\cdots,\nu\left(L(c_n)\right)\right]. \tag{2.7}$$

其中 $\underline{T}$ 為經過轉換後的 tanh 域向量。接下來，利用式 (2.7) 定義 $\mathbf{H_b}$ 中第 j 個校驗方程式 (即 $\mathbf{H_b}$ 中的第 j 列) 的決策信任度 γ_j (decision reliability) 於式 (2.8) 中：

$$\gamma_j=\prod_{p\in\Gamma_j}^{n}\nu\left(L(c_p)\right). \tag{2.8}$$

其中 $\Gamma_j\in\left\{a\,|\,\forall 1\le a\le n,\ H_{bja}=1\right\}$。$\gamma_j$ 於此所代表的意義有兩項 (1) 若將 LLR 經過硬式決策後，則可計算 LLR 相對於 $\mathbf{H_b}$ 的徵狀 (syndrome) 。當 $\gamma_j>0$ 時，表示徵狀中第 j 個位置為 0。若 $\gamma_j<0$，則表示徵狀中第 j 個位置為 1。 (2) 依一般信任度解碼的思考，若 $\mathbf{H_b}$ 中第 j 個校驗方程式所校驗到的位元位置 (即 $\mathbf{H_b}$ 的第 j 列於該位置之值為 1)，其 LLR 的絕對值較大時，會認為該位置的硬式決策較不易發生錯誤，若所校驗到位元位置其 LLR 之絕對值較小時，會認為該位元位置的硬式決策較易發生錯誤。同樣的概念應用於此處的決策信任度上，當 $|\gamma_j|$ 愈大時，會認為徵狀中第 j 個位置的決策(為 1 或 0)較不易發生錯誤，而當 $|\gamma_j|$ 愈小時，會認為該決策較易發生錯誤。由式 (2.5) 和式 (2.6) 可知，γ_j 必定界於

$[-1,+1]$ 間。因此若 LLR 經過 $\mathbf{H_b}$ 中第 j 個校驗方程式校驗為一合法碼字，則 $\gamma_j > 0$，且 $|\gamma_j|$ 愈大表示我們能夠愈信任這個校驗的決策，反之 $|\gamma_j|$ 愈小，則表示這個校驗的決策愈容易發生錯誤。

接下來以式 (2.9) 定義一位能函數 (potential function) J

定義 2.1： 定義位能函數 J 為

$$J(\mathbf{H_b}, \underline{T}) = -\sum_{j=1}^{(n-k)} \gamma_j = -\sum_{j=1}^{(n-k)} \prod_{p \in \Gamma_j} T_p \,. \tag{2.9}$$

其中 $\Gamma_j \in \{a \mid \forall 1 \le a \le n,\ H_{bja} = 1\}$，且 J 是一個同時受 $\mathbf{H_b}$ 和 $\underline{T}$ 所影響的函數。

由以上的討論可知，若一個校驗方程式對於 LLR 的校驗結果一定為正確時，我們會認為其 $|\gamma_j|$ 為 1。而當 $\gamma_j > 0$ 且 $|\gamma_j| = 1$ 時，則表示我們認為該校驗方程式對 LLR 所做的校驗結果必定徵狀為 0，也就是該校驗方程式認為其一定為合法的碼字。

以一個合法的碼字而言，其必定滿足所有的校驗方程式，而徵狀必定均為 0。因此在校驗合法碼字的情況下，每個校驗方程式的 γ_j 必定為 1，且此時所有校驗方程式的 γ_j 總合為 $(n-k)$。由式 (2.9) 可知此時位能函數值 J 會到達最小值 $-(n-k)$。故若能由通道所得到信任度 $\underline{T}$ 為初始起點，使接收信號的 LLR 往位能函數最低點的方向前進，並收斂於位能函數的最低點時，則被認為此過程成功找到了傳送的碼字而成功解碼。

因此基於這樣的想法，梯度下降法適合進行這樣的運作過程。 首先對位能函數進行偏微分後，得到式 (2.10)

$$\nabla J(\mathbf{H_b}, \underline{T}) = \left(\frac{\partial J(\mathbf{H_b}, \underline{T})}{\partial T_1}, \frac{\partial J(\mathbf{H_b}, \underline{T})}{\partial T_2}, \ldots, \frac{\partial J(\mathbf{H_b}, \underline{T})}{\partial T_n} \right). \tag{2.10}$$

其中每個對應T_i的偏微分元素可以表示成式 (2.11) 的形式

$$\frac{\partial J\left(\mathbf{H_b}, \underline{T}\right)}{\partial T_i} = -\sum_{j\in\Lambda_i}\prod_{p\in\Omega_j} T_p \,. \tag{2.11}$$

此時$\Lambda_i = \left\{q \,|\, \forall 1 \le q \le n-k,\, H_{bqi} = 1\right\}$且$\Omega_j = \left\{s \,|\, \forall 1 \le s \le n,\, H_{bjs} = 1,\, s \ne i\right\}$。

最後以式 (2.12) 來描述梯度下降法搜尋位能函數最低點的過程

$$\underline{T}^{(l)} - \alpha \nabla J\left(\mathbf{H_b}, \underline{T}^{(l)}\right) \to \underline{T}^{(l+1)} \,. \tag{2.12}$$

這裡α為一抑制係數，如同梯度下降法中控制每次疊代的步距 (step size) 係數，上標(l)表示目前為第l次疊代。由於信任度 T_i 以 tanh 域表示時，其被定義在$[-1,+1]$的區間內，因此需修正原來的梯度下降法的疊代函數，以保證信任度T_i始終於$[-1,+1]$的區間內。經過最後經過修正後，可得到式 (2.13) 的形式。

$$\nu^{-1}\left(T_i^{(l)}\right) - \alpha\left[-\sum_{j\in\Lambda_i}\nu^{-1}\left(\prod_{p\in\Omega_j} T_p^{(l)}\right)\right] \to \nu^{-1}\left(T_i^{(l+1)}\right). \tag{2.13}$$

此時$\Lambda_i = \left\{q \,|\, \forall 1 \le q \le n-k,\, H_{bqi} = 1\right\}$且$\Omega_j = \left\{s \,|\, \forall 1 \le s \le n,\, H_{bjs} = 1,\, s \ne i\right\}$，$T_p^{(l)}$和$T_i^{(l)}$之上標$(l)$表示目前為第$l$次疊代。

實際運作上，由式 (2.5) 和 (2.13) 式，每個位置的 LLR ($L(c_i)$)在第l次疊代過程中所接收到的額外消息量 (extrinsic information) 總值$L_{ext}^{(l)}(c_i)$可以表示如式 (2.14) 所示：

$$L_{ext}^{(l)}(c_i) = \sum_{j\in\Lambda_i}\nu^{-1}\left(\prod_{p\in\Omega_j} T_p^{(l)}\right) = \sum_{j\in\Lambda_i} 2\tanh^{-1}\left(\prod_{p\in\Omega_j}\tanh\left(\frac{L^{(l)}(c_p)}{2}\right)\right). \tag{2.14}$$

其中$\Lambda_i = \left\{q \,|\, \forall 1 \le q \le n-k,\, H_{bqi} = 1\right\}$且$\Omega_j = \left\{s \,|\, \forall 1 \le s \le n,\, H_{bjs} = 1,\, s \ne i\right\}$，$T_p^{(l)}$和$L^{(l)}(c_p)$之上標$(l)$表示目前為第$l$次疊代。而由式 (2.14) 可知，額外消息量總值中，由第j個校驗方程式所提供的消息量可用式 (2.15) 來表示，這裡以上標

$(l)(j)$ 來表示額外消息量 $L_{ext}^{(l)(j)}(c_i)$ 由第 j 個校驗節點所提供。

$$L_{ext}^{(l)(j)}(c_i) = 2\tanh^{-1}\left(\prod_{p\in\Omega_j}\tanh\left(\frac{L^{(l)}(c_p)}{2}\right)\right). \tag{2.15}$$

其中 $\Omega_j = \{s \mid \forall 1 \le s \le n,\ H_{b\,js} = 1,\ s \ne i\}$ 。

2-3.3 梯度下降法解碼步驟

經由前一節之介紹，接下來我們將梯度下降法的解碼執行步驟陳述如下：

步驟 1、 設定抑制係數 α， 最大疊代次數 $l_{\max}$，並計算接收信號中每個元素的初始的 LLR 為 $L^{(0)}(c_i) = (4y_i / N_0)$，並以初始 LLR ($L^{(0)}(c_i)$) 開始進行解碼。

步驟 2、 利用梯度下降法於每次疊代中，計算每個位元位置所得到的額外消息量 $L_{ext}^{(l)}(c_i)$，計算方式前一小節之式 (2.14)。

步驟 3、 經由方程式 $L^{(l+1)}(c_i) = L^{(l)}(c_i) + \alpha L_{ext}^{(l)}(c_i)$，將每個位元位置的 LLR ($L^{(l)}(c_i)$) 進行更新，此過程即前一小節中式 (2.13) 的運作。解碼過程中，若已達預設的最大疊代次數 $l_{\max}$，或著疊代過程中經硬式決策後能得到一個合法碼字時，則停止解碼並輸出目前的解碼結果。若未達預設最大疊代次數且未得到合法碼字時，則以更新後的 LLR ($L^{(l+1)}(c_i)$)，回到步驟 2 進行下一次疊代。

2.4 梯度下降法之圖形化運作方式

經過二位元映射後的奇偶校驗矩陣均可用以下的 Tanner 圖來表示，如

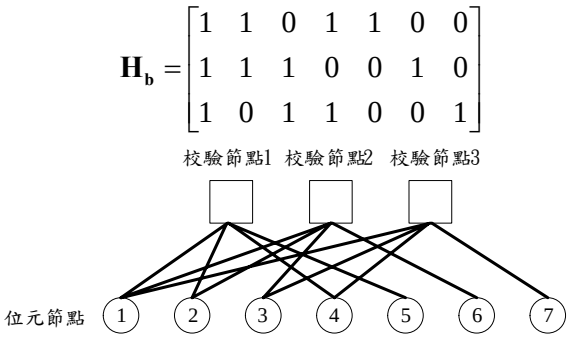


圖 2-1 使用 Tanner 圖表示二位元奇偶校驗矩陣

$\mathbf{H_b}$ 上的每一列，在 Tanner 圖上是使用校驗節點 (check node) 表示；而碼字上的每個元素 c_i，在 Tanner 圖上則是用相對應的位元節點 (bit node) 來表示。當 $\mathbf{H_b}$ 中的第 i 列第 j 行的位置為 1 時，則將則第 i 列所代表的校驗節點和第 j 行的位元節點在圖形上以相連來表示。則原有的梯度下降法對應至 Tanner 圖上時，可利用簡單的圖形化動作來表示其運作過程。

圖 2-2 表示梯度下降法的初始過程，即前一小節中的步驟一。每個位元節點均可經由接收信號得到一初始的 LLR 值：$L^{(0)}(c_i) = (4y_i / N_0)$，並且傳送給相連的校驗節點。

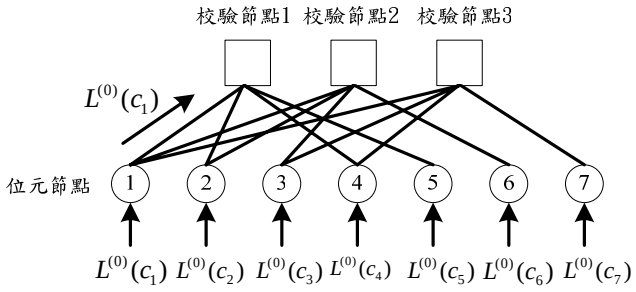


圖 2-2 給予每個位元節點初始 LLR 示意圖

圖 2-3 為梯度下降法計算第 l 次疊代時，校驗節點提供位元節點額外消息量的表示圖，即式 (2.15) 的過程。以校驗節點 1 為範例，計算其給予位元節點 5 之額外消息量時，其消息來源分別來自位元節點 1、位元節點 2 和位元節點 4。

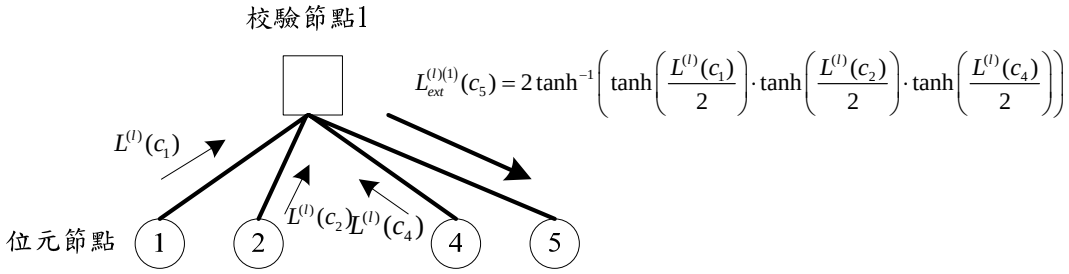


圖 2-3 計算校驗節點給予位元節點之額外消息量示意圖

圖 2-4 則為梯度下降法更新位元節點 LLR 的動作，即式 (2.14) 的過程。以位元節點 1 為範例，其於該次疊代中得到校驗節點 1、校驗節點 2 和校驗節點 3 所提供的額外消息量 $L_{ext}^{(l)(1)}(c_i)$ 、 $L_{ext}^{(l)(2)}(c_i)$ 和 $L_{ext}^{(l)(3)}(c_i)$ 。

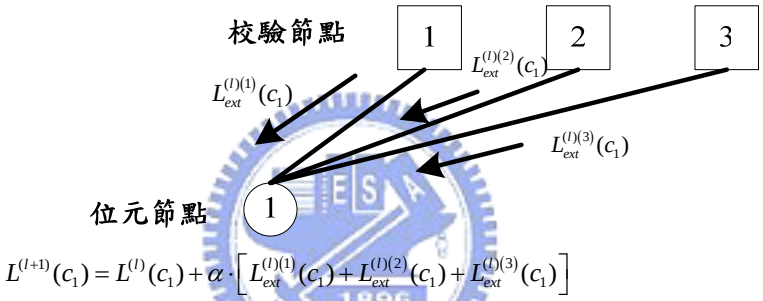


圖 2-4 更新每個位元之 LLR 值示意圖

而梯度下降法和傳統的和積演算法 (sum-product algorithm) 不同之處在於位元節點給予校驗節點消息量時，梯度下降法由同一個位元節點給予其相連所有校驗節點的消息量均為 $L^{(l)}(c_i)$ ，而和積演算法則是由圖 2-5 的方式計算給予校驗節點消息量，其中 $Q_{(1)}(c_1)$ 表示由第 1 個位元節點給予第一個校驗節點之消息量。而在計算位元節點給予校驗節點之消息量時，其中由位元節點本身所提供的 LLR 值，在梯度下降法時為前一次疊代時位元節點的 LLR ($L^{(l)}(c_i)$)，在和積演算法時則均為該位元節點的初始 LLR ($L^{(0)}(c_i)$)。

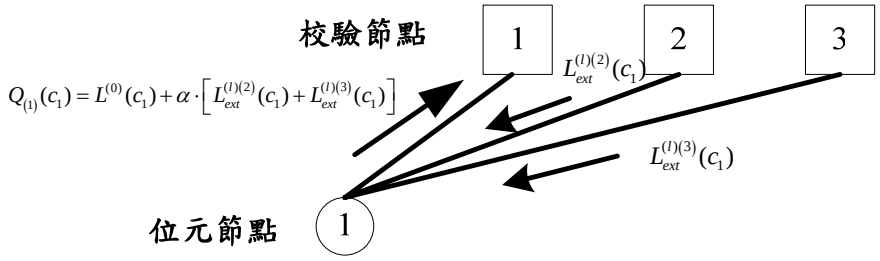


圖 2-5 和積演算法由位元節點給予校驗節點消息量示意圖

在本研究接下來的數章中，將會利用到以上圖形化表示的方式來方便說明梯度下降演算法相關的應用。

第三章

JN 解碼演算法及 ABP-OSD(1)解碼演算法

本章將介紹 JN 解碼演算法和 ABP-OSD(1) 解碼演算法的解碼方式。

3-1 JN 解碼演算法

3-1.1 JN 解碼演算法簡介

首先我們將里德所羅門碼經由二位元映射後得到二位元奇偶校驗矩陣 $\mathbf{H}_b$，再利用此 $\mathbf{H}_b$ 進行接下來的解碼動作。

解碼過程中，每次的疊代動作主要可分為兩個階段如下：

(1) 更新奇偶校驗矩陣階段

於此階段，首先將所有位元節點的 LLR 依照其信任度排序，其後按照信任度由小至大的對應位置進行高斯消去法，使 $\mathbf{H}_b$ 成為一簡化梯式(row reduced echelon form) 矩陣。此動作由最小信任度的位置開始，利用列運算將該位置正規化後，再對信任度次高的位置進行同樣的動作，若發現該位置無法正規化，則略過該位置進行信任度次高的下一個位置，直到得到 $\mathbf{H}_b$ 的簡化梯式矩陣 $\mathbf{H}_\mathbf{b}'$ 和 $(n-k)$ 個成功正規化的位置，稱為最小信任獨立位置(least reliable independent positions，簡稱 LRIPs)，這些位置可以經由重新排序而成為一正規化矩陣 (identity matrix)。而非最小信任獨立位置的部分於本論文中則稱為最大信任度位置 (most reliable positions ，簡稱 MRPs) ，由於 $\mathbf{H}_\mathbf{b}'$ 中對應到這些 LRIPs 的行向量，均只有一個 1，因此這些部分亦稱為 $\mathbf{H}_\mathbf{b}'$

的低密度位置，反之這些位置以外則稱為 $\mathbf{H}_b'$ 的高密度位置，如圖 3-1 所示。

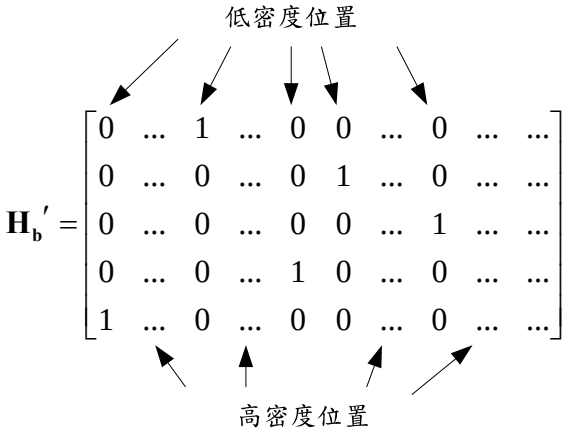


圖 3-1 經高斯消去法後，二位元奇偶校驗矩陣內的低密度位置及高密度位置

(2) 更新位元節點之 LLR 階段

接下來利用經過高斯消去法作用後的簡化梯式矩陣 $\mathbf{H}_b'$，使用梯度下降法求得每個位元節點的額外消息量後更新原有的 LLR，即前式 (2.13) 之動作。

Jing Jiang 和 Narayanna 於 [15]中提出，一般的梯度下降法運作時，若一個校驗節點相連至多個低信任度的位元節點，則受這些低信任度的位元節點的影響，校驗節點傳送給位元節點的訊息量大部分會近似為 0，因此造成梯度下降法無法提供位元節點足夠的額外消息量以更正錯誤，此現象稱為校驗節點飽和化 (check node saturation) ，如圖 3-2 所示。因此使用 JN 解碼演算法，則可使一個校驗節點僅連接至一個低信任度的位元節點，減少校驗節點飽和化所帶來的影響，以增進其效能。

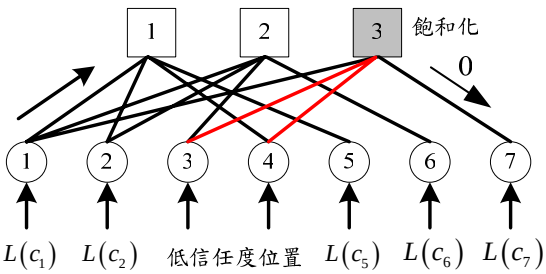


圖 3-2 過多低信任度位置造成梯度下降演算法無法提供位元節點保護能力

3-1.2 JN 演算法解碼步驟

於本節中，將 JN 演算法執行的詳細步驟說明如下：

步驟一、 設定抑制係數 α， 最大疊代次數 $l_{\max}$，並計算每個位元節點的初始的 LLR 為 $L^{(0)}(c_i)=\left(4y_i/N_0\right)$，並以初始 LLR ($L^{(0)}(c_i)$) 開始進行解碼。

步驟二、更新奇偶校驗矩陣：$\mathbf{H}_b^{(l)}=\phi\left(\mathbf{H}_b,\left|\underline{L}^{(l)}\right|\right)$，$\phi$ 表示更新 $\mathbf{H}_b$ 的動作。

(a) 依據 LLR 絕對值 $\left|\underline{L}^{(l)}\right|$ 的大小將每個位元位置進行排序，並且記錄排序後的順序。

(b) 利用高斯消去法，按照信任度由小至大的位置將 $\mathbf{H}_b^{(l)}$ 消為簡化梯式矩陣，並得到 $(n-k)$ 個 LRIPs。

步驟三、 利用高斯消去法正規化後的 $\mathbf{H}_b^{(l)}$，進行梯度下降法得到每個位元節點的額外消息量：$\underline{L}_{ext}^{(l)}=\varphi\left(\mathbf{H}_b^{(l)},\underline{L}^{(l)}\right)$，即式 (2.14) 的過程。$\varphi$ 表示計算額外消息量之動作。

步驟四、 更新每個位元節點的 LLR： $\underline{L}^{(l+1)}=\underline{L}^{(l)}+\alpha\underline{L}_{ext}^{(l)}$，即式 (2.13) 的過程。在這裡 $0<\alpha\le 1$。

步驟五、 硬式決策：$\hat{c}_i=\begin{cases}0, & L^{(l+1)}(c_i)>0\\ 1, & L^{(l+1)}(c_i)<0\end{cases}$

步驟六、 終止條件：若所有的校驗節點均被滿足，或當到達最大的預設疊代次數時，則停止疊代並且輸出目前經硬式決策後每個位元的決策值。若未達最大疊代次數，則令 $l=l+1$，回到步驟二進行下一次的疊代。

3-1.3 JN 演算法解碼範例

使用(7,4)漢明碼作為範例，原始的二位元奇偶校驗矩陣 $\mathbf{H}_b$ 和相對應每個位

元位置的初始 LLR 如圖 3-3 所示。

$$\mathbf{H_b} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

位元1	位元2	位元3	位元4	位元5	位元6	位元7
1.2	-0.01	10.0	4.3	-5.2	-6.6	-7

圖 3-3 JN 演算法之初始化範例

首先經過排序，由小至大的順序分別為圖 3-4 所示。

位元1	位元2	位元3	位元4	位元5	位元6	位元7
1.2	-0.01	10.0	4.3	-5.2	-6.6	-7
順序2	順序1	順序7	順序3	順序4	順序5	順序6

圖 3-4 JN 演算法之排序範例

接下來將二位元奇偶校驗矩陣，依信任度排序由小至大進行高斯消去動作後，得到對應的簡化梯式矩陣 $\mathbf{H_b}'$、LRIP 的位置和相對應的 Tanner 圖，如圖 3-5 所示。

$$\mathbf{H_b}' = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

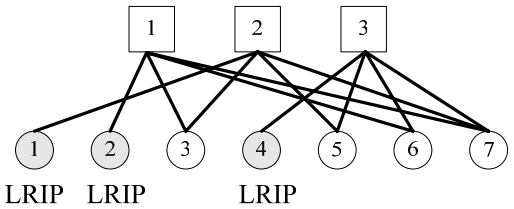
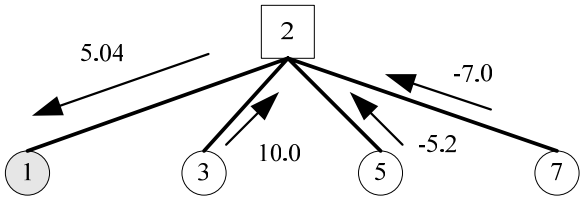


圖 3-5 JN 演算法之高斯消去動作範例

接下來利用經高斯消去動作後的簡化梯式矩陣 $\mathbf{H_b}'$ 進行梯度下降演算法，並得到每個位元節點的額外消息量，更新原有之 LLR，如圖 3-6 所示。

位元節點1之額外消息量計算和LLR更新



$$L_{ext}^{(l)(2)}(c_1) = 2\tanh^{-1}\left(\tanh\left(\frac{L^{(l)}(c_3)}{2}\right)\cdot\tanh\left(\frac{L^{(l)}(c_5)}{2}\right)\cdot\tanh\left(\frac{L^{(l)}(c_7)}{2}\right)\right)$$

$$= 2\tanh^{-1}\left(\tanh\left(\frac{10.0}{2}\right)\cdot\tanh\left(\frac{-5.2}{2}\right)\cdot\tanh\left(\frac{-7.0}{2}\right)\right)$$

$$= 5.04$$

$$L^{(l+1)}(c_1) = L^{(l)}(c_1) + \alpha\left[L_{ext}^{(l)(2)}(c_1)\right] = 1.2 + 1\cdot[5.04]$$

此時 α 預設為1，如以上之步驟，可計算所有位元節點之額外消息量和並進行LLR更新

	位元1	位元2	位元3	位元4	位元5	位元6	位元7
原始的LLR	1.2	-0.01	10.0	4.3	-5.2	-6.6	-7.0
	位元1	位元2	位元3	位元4	位元5	位元6	位元7
額外消息量	5.04	6.07	1.17	-4.86	2.95	3.92	2.72
	位元1	位元2	位元3	位元4	位元5	位元6	位元7
更新後的的LLR	6.24	6.06	11.17	-0.56	-2.15	-2.68	-4.28

圖 3-6 JN 演算法之 LLR 更新範例

接下來以更新後的 LLR 進行下一次的疊代過程。

3-1.4 JN 解碼演算法之改善方式

Jing Jiang 和 Narayanna 於[15]提出了三種改進原有 JN 演算法效能的方式，接下來將介紹此三種方法。

(1) 階段-2 隨機相連方式

由於經過高斯消去法作用後，位於$\mathbf{H}_b$低密度位置的位元節點均只相連至一個校驗節點，因此其只會收到該校驗節點所傳送的額外消息量。若該校驗節點相連的其他高密度位置之位元節點發生錯誤時，則此低密度位置的位元節點容易收到錯誤的額外消息量而翻轉其 LLR 的正負號，整個解碼過程也容易受此影響而收斂到錯誤的碼字。因此提出以下的方式，於更新奇偶校驗矩陣階段修正這個問題。

階段-2 隨機相連演算法

步驟一、 按照原有的方式對每個位元節點的 LLR 進行排序，並且由信任度小至大進行高斯消去法得到$\mathbf{H}_b$的低密度位置和高密度位置。

步驟二、 將$\mathbf{H}_b$的列 1 至列 $(n-k)$ 間進行隨機的排序，並記錄排序後的索引值為$p_1, p_2, p_3, ..., p_{n-k}$。

如 $[1 \quad 2 \quad 3 \quad 4] \rightarrow [2 \quad 4 \quad 1 \quad 3] = [p_1 \quad p_2 \quad p_3 \quad p_4]$。

步驟三、 將高斯消去法後得到的$\mathbf{H}_b$，按照以下的方式進行修正：

將列p_{i+1}加至列p_i，自$i=1$至$i=n-k-1$。

經過此演算法作用後，除了p_1行仍然只相連至一個校驗節點外，所有的低密度部分的位元節點均相連至兩個校驗節點，而抑制了上述問題的影響。

(2) 平行選擇不同的位元節點作為低密度位置

按照 LLR 排序進行高斯消去法後，某些高密度位置的位元節點，其信任度相當接近低密度位置的位元節點，而這些高密度部分的位元節點實際上由於信任度不大，因此也較容易發生錯誤的情況，因此在 [15] 中針對此種情況提出了改進的方式。

其平行進行數個原有的 JN 演算法，然而每個 JN 演算法於一開始均使用不

同低密度位置的位元節點，也就是將部分位於高信任度和低信任度交界處的低信任度位置和高信任度位置進行交換，再依交換後的順序進行高斯消去法和梯度下降法的疊代過程。這些平行處理的數個 JN 演算法由於初始時為不同的排序而在初始時有不同的 $\mathbf{H}_b$。每個 JN 演算法於每次疊代中均會經過硬式決策而得到一組解碼輸出，假設經過 l 次疊代，則會產生 l 個解碼輸出，若現在共有 A 組 JN 演算法使用不同的排序平行處理，則總共會有 Al 個解碼輸出，則在這 Al 個解碼輸出中，最後將選擇和接收信號相關性最高的解碼輸出作為最終的解碼輸出。

(3) 合併代數解碼 (algebraic decoding) 方式

此方式為對 JN 演算法每次疊代的解碼輸出再予以進行代數解碼，因此在每次疊代中解碼器均會得到一個代數解碼的輸出。由於代數解碼所輸出的碼字有可能非為與接收信號相關性最大的碼字，因此 JN 演算法在每次解碼時，將執行至最大的疊代次數後，選擇代數解碼輸出的所有碼字中和接收信號間相關性最大的碼字做為最後的解碼輸出。

在此特別說明，以下於本研究中所提及之 JN 演算法，皆為原始之 JN 演算法，並未使用以上三種方式。

3-2 ABP-OSD(1) 解碼演算法

3-2.1 ABP-OSD(1) 解碼演算法簡介

ABP-OSD(1) 解碼演算法[12]結合了兩種使用信任度排序的解碼法。其中 ABP 的過程是利用二位元奇偶校驗矩陣由信任度小至大的位置進行高斯消去法得到 LRIPs 和相對應的簡化梯式奇偶校驗矩陣後，再計算每個位元節點額外消息量的動作。OSD(1) 則是利用二位元生成矩陣由信任度大至小的位置進行高斯消去法得到 k 個最大信任獨立位置 (most reliable independent positions，簡稱 MRIPs)

和簡化梯式生成矩陣後，再以這些位置配合測試錯誤樣本 (test error patterns，以下簡稱為 TEPs) 重新編碼來來進行解碼。TEP 為長度 k 的向量，若預設第 i 個 MRIP 為錯誤的位置時，TEP 的該位置即為 1，否則為 0，並以 w 來表示 TEP 中 1 的個數。我們可以利用 TEP 加上 MRIPs 來獲得更新的 MRIPs 後，再利用更新後的 MRIPs 重新編碼，若錯誤的確均發生在 TEP 所預測的位置上時，則可以編碼出正確的碼字。以下關於 OSD 的討論中，碼字的非 MRIPs 部分，稱為最小信任位置 (least reliable positions，簡稱 LRPs)。於下一節中，我們將說明 OSD 之運作方式。

3-2.2 OSD (w) 解碼演算法步驟

於說明 OSD(w)之解碼步驟前，我們先介紹解碼中所使用到的判斷原則和數學表示，其相關說明如下：

首先由碼字 $\underline{c}$ 和接收信號定義兩個集合於式 (3.1) 和式 (3.2)：

$$D_0(\underline{c}) \triangleq \{i : c_i = z_i, 1 \le i \le n\} \tag{3.1}$$

$$D_1(\underline{c}) \triangleq \{i : c_i \neq z_i, 1 \le i \le n\}. \tag{3.2}$$

其中 z_i 為接收信號經硬式決策後所得的 0 或 1 之值。

另外定義 $D_1(\underline{c})$ 中的個數於式 (3.3)：

$$n(\underline{c}) = |D_1(\underline{c})|. \tag{3.3}$$

接下來定義接收信號和碼字間的相關性差異 (correlation discrepancy) 為 λ，而其計算方式為式 (3.4)。

$$\lambda(\underline{r}, \underline{c}) = \sum_{i \in D_1(\underline{c})} |r_i|. \tag{3.4}$$

而接收信號和碼字間的相關性 (correlation) θ 可定義為式 (3.5)

$$\theta(\underline{r}, \underline{c}) \triangleq \sum_{i=1}^{n} r_i \cdot (1 - 2c_i). \tag{3.5}$$

在 $D_0(\underline{c})$ 中，信任度最小的前 $d_{\min}-n(\underline{c})$ 個位置，我們以集合 $D_0^A(\underline{c})$ 來表示。

若 $d_{\min}<n(\underline{c})$ 時，則定義 $D_0^A(\underline{c})$ 為空集合。

由以上的定義，我們最後定義一比較係數 $G(\underline{c})$ 於式 (3.6)

$$G(\underline{c}) \triangleq \sum_{i\in D_0^A(\underline{c})} |r_i|. \tag{3.6}$$

由[16]可知，當式 (3.7) 成立時，則表示 $\underline{c}$ 為和接收信號相關性 θ 最大的碼字，即最大概似碼字。

$$\lambda(\underline{r},\underline{c}) \le G(\underline{c}). \tag{3.7}$$

以下為 OSD(w)演算法之詳細執行步驟，運作過程可以圖 3.7 表示：

步驟一、 依據信任度大小進行位元節點的排序。

步驟二、 依排序的結果，依信任度由大至小的位置對二位元生成矩陣 $\mathbf{G}_b$ 進行高斯消之動作而得到簡化梯式生成矩陣。若該位置無法經列運算正規化，則跳過該位置至信任度次大的下一個位置，直到得到 k 個 MRIPs 後停止。

步驟三、 製作測試錯誤樣本：假設此 k 個 MRIPs 中分別有 0、1、2、…….、w 個錯誤，($w < k$)，此於演算法之標示為 OSD(w)，則表示此演算法於該 MRIPs 中，最多測試至發生 w 個錯誤。接下來分別產生 0、1、2、…….、w 個錯誤的所有測試錯誤樣本，若該位置發生錯誤，則在樣本中給予 1 的值，否則給 0 的值，而在 OSD(w)的情況下，總共產生 $\sum_{i=0}^{w}\binom{k}{i}$ 個 TEPs。

步驟四、 將原來的 MRIPs 向量加上 TEP 向量後，根據高斯消去法所得到的梯式簡化生成矩陣進行編碼，當所有的 TEP 配合 MRIPs 均被重新編碼後，選擇和接收信號相關性最大碼字的做為解碼輸出。或當某個 TEP

配合 MRIPs 所得到的碼字為最大概似碼字時，則直接輸出該碼字。

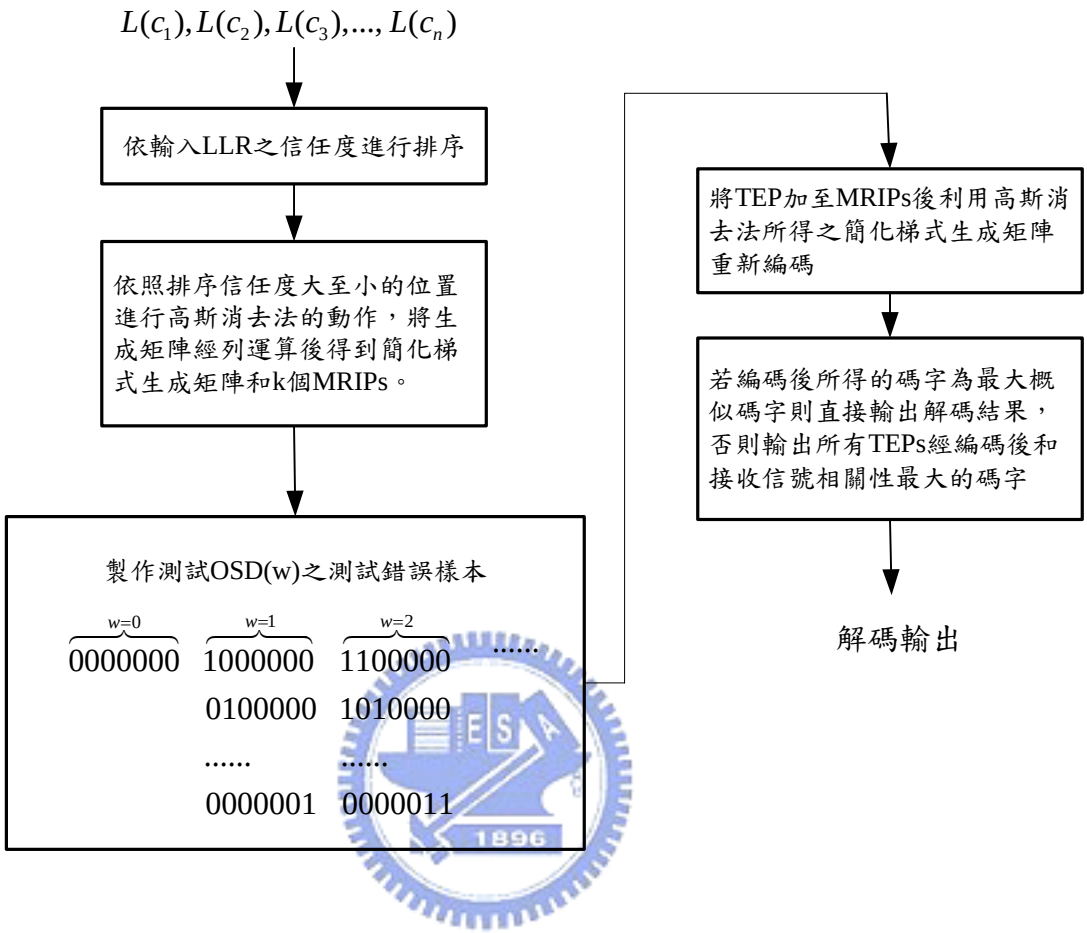


圖 3.7 OSD(w) 演算法運作過程

而在 ABP-OSD(1)解碼演算法中，使用 w 為 1 的情況。

3-2.3 OSD(1) 解碼演算法範例

使用(7,4)漢明碼作為範例，原始的二位元生成矩陣和相對應每個位元位置的初始 LLR 如下：

$$\mathbf{G_b} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

位元1	位元2	位元3	位元4	位元5	位元6	位元7
1.2	-0.01	10.0	4.3	-5.2	-6.6	-7

圖 3-8 OSD(1)演算法之初始化範例

首先經過排序

位元1	位元2	位元3	位元4	位元5	位元6	位元7
1.2	-0.01	10.0	4.3	-5.2	-6.6	-7
順序2	順序1	順序7	順序3	順序4	順序5	順序6

圖 3-9 OSD(1)演算法之排序範例

則將生成矩陣由高信任度位置由大至小開始做高斯消去動作後，得到對應的簡化梯式生成矩陣和最大獨立信任度位置，如圖 3-11

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

MRIPs

圖 3-10 OSD(1)演算法之初始化範例

將 LLR 經過硬式決策後，得到每個位置的決策值，接下來設定 $k+1$ 個 TEPs，加至決策值後的向量，再將此向量經由高斯消去動作後的生成矩陣重新編碼產生碼字，如圖 3-11。

$MRIP = 0111$ 例

TEP	$= 0000$
	0001
	0010
	0100
	1000

$MRIP + TEP = 0111 + 0001 = 0110$

重新編碼後得到碼字

$$MRIP \bullet G' = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \bullet \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

圖 3-11 OSD(1)演算法之重新編碼範例

最後計算該碼字和接收信號間的相關性，若其為最大概似碼字則直接輸出，否則保留目前相關性最大的碼字，最後以相關性最大的碼字作為最終的解碼輸出。

3-2.4 JN-OSD(1) 解碼演算法步驟

於 ABP-OSD(1)演算法中，本研究於 ABP 的部分使用以梯度下降法為基礎的 JN 演算法，以下簡稱為 JN-OSD(1)解碼演算法。

JN-OSD(1)解碼演算法的詳細步驟如下：

步驟一、設定抑制係數 α， 最大疊代次數 $l_{\max}$，並計算每個位元節點的初始的 LLR 為 $L^{(0)}(c_i) = (4y_i / N_0)$，並以初始 LLR ($L^{(0)}(c_i)$) 開始進行解碼。

步驟二、OSD(1)演算法階段：

(1) 依據 LLR 絕對值 $\left|\underline{L}^{(l)}\right|$ 的大小將每個位元位置進行排序，並且記錄排序後的順序。

(2) 依排序的結果，依信任度由大至小的位置對二位元生成矩陣 $\mathbf{G}_b$ 進行高斯消之動作而得到簡化梯式生成矩陣。若該位置無法經列運算正規化，則跳過該位置至信任度次大的下一個位置，直到得到 k 個 MRIPs 後停止。

(3) 此時假設此 k 個 MRIPs 中僅有一個錯誤，因此生成 w 為 0 和 w 為 1 的 $k+1$ 個 TEPs。並按照此 $k+1$ 個 TEPs 以高斯消去動作後的生成矩陣進行編碼得到 $k+1$ 組候選碼字 (candidate codewords)。

(4) 若找到最大概似碼字，則停止解碼並輸出結果。若到達最大預定疊代次數 $l_{\max}$ 時，則停止解碼並輸出最目前所存相關性最大的碼字。未停止解碼則進行步驟三。

步驟三、 JN 演算法階段。

(1) 更新奇偶校驗矩陣：$\mathbf{H}_b^{(l)} = \phi\left(\mathbf{H}_b, \left|\underline{L}^{(l)}\right|\right)$，$\phi$ 表示更新 $\mathbf{H}_b$ 的動作。利用高斯消去法，按照信任度由小至大的位置將 $\mathbf{H}_b^{(l)}$ 消為簡化梯式矩陣，並得到 $(n-k)$ 個 LRIPs。

(2) 利用高斯消去法正規化後的 $\mathbf{H}_b^{(l)}$，進行梯度下降法得到每個位元節點的額外消息量：$\underline{L}_{ext}^{(l)} = \varphi\left(\mathbf{H}_b^{(l)}, \underline{L}^{(l)}\right)$，即式 (2.14) 的過程。$\varphi$ 表示計算額外消息量之動作。

(3) 更新每個位元節點的 LLR：$\underline{L}^{(l+1)} = \underline{L}^{(l)} + \alpha \underline{L}_{ext}^{(l)}$，即式 (2.13) 的過程。在這裡 $0 < \alpha \leq 1$。

(4) 回到步驟二。

ABO-OSD(1)之機制可以用圖 3-12 來表示

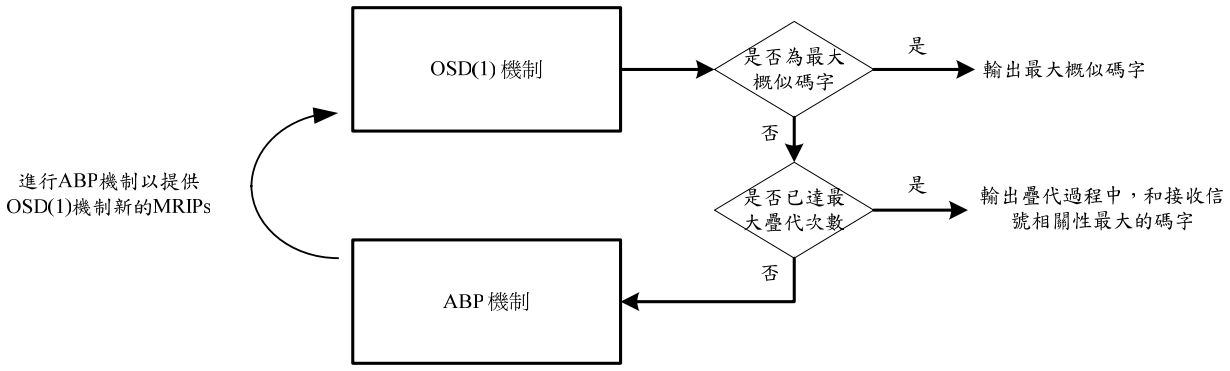


圖 3-12 ABP-OSD(1) 機制運作方式

第四章

抑制 JN 演算法中高信任度錯誤之演算法

於本章中首先將指出原有 JN 演算法的潛在問題，並說明本研究中如何於提出之演算法中改善此潛在問題，並且佐以數據分析以證明此潛在問題受到了抑制。於最後一節中，再提出利用里德所羅門碼的內插特性，輔助解碼器改善效能之概念。

4-1 JN 演算法之探討

4-1.1 JN 演算法之優勢

於信念傳遞解碼中，通常希望錯誤不要發生在二位元奇偶校驗矩陣的高密度位置中，以免由於錯誤相連至過多的校驗節點，而造成錯誤傳遞的問題。基於信任度解碼的概念，在 AWGN 通道之下，接收信號的 LLR 絕對值 (信任度) 愈小的位置，會被認定較有可能發生錯誤。因此於 JN 演算法中，其利用高斯消去得到 LRIPs 的動作，使得每個低信任度位元節點僅相連至一個校驗節點，每個校驗節點也只相連至一個低信任度的位元節點。這個動作除了於前一章所提到防止校驗節點飽和化的作用外，也同樣的抑制了低信任度錯誤所造成的錯誤傳遞問題。接下來我們利用以下的分析方式來說明這個現象。

若現有二位元奇偶校驗矩陣之 Tanner 圖如圖 4-1 所示，圖中週期 4 (cycle 4) 的部分所代表的是二位元奇偶校驗矩陣中的高密度位置。在以下的討論中，假設

正確的 LLR 值為一正值 (+)，而錯誤的 LLR 值為一負值 (-)。我們假設在沒有發生錯誤的情況下，高信任度位置 LLR 的絕對值為 V ，低信任度位置的 LLR 的絕對值為 v，且 V>v。若高信任度位置發生錯誤，則我們令其 LLR 值為-V，若低信任度位置發生錯誤，我們令其 LLR 值為-v。在圖 4-1 中，我們同時假設位元節點 3 和位元節點 4 為低信任度位置，其他位元節點為高信任度位置。

接下來我們假設為低信任度位於高密度位置的位元節點 3 發生錯誤並給予其-v 的 LLR 值，如圖 4-2 所示。利用梯度下降法的運作，每個位元節點可分別自校驗節點 1 或校驗節點 2 得到計算後的額外消息量。由於目前假設正確的 LLR 為一正值，因此當校驗節點給予某個位元節點正確的額外消息量時，此額外消息量我們以+E (E>0) 來表示，而當校驗節點給予某個位元節點錯誤的額外消息量時，此額外消息量則以-E 來表示。

因此利用以上的假設，我們可以得到每個位元節點所收到的額外消息量，以及更新後的 LLR 值，如圖 4-3 所示。從訊息的傳遞過程中發現，圖 4-3 中除了錯誤的位元節點外，所有其他的位元節點均接收到了錯誤的額外消息量。這種錯誤額外消息量的大量傳播容易造成其他位元的錯誤。因此若錯誤發生於高密度位置時，往往造成梯度下降法收斂至錯誤的碼字，甚至無法收斂的現象。

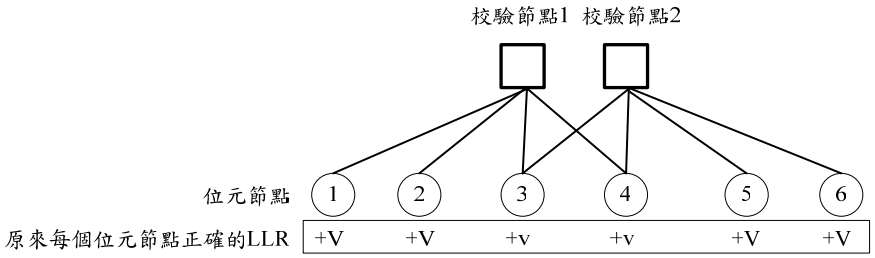


圖 4-1 位元節點 3 和位元節點 6 為低信任度位置

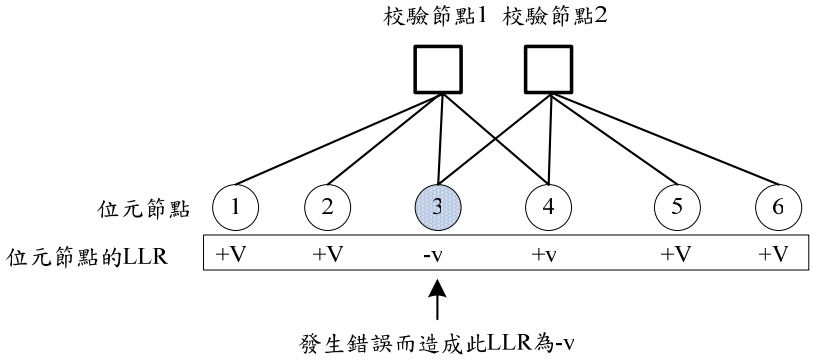


圖 4-2 當高密度位置 (位元節點 3) 發生一個錯誤時

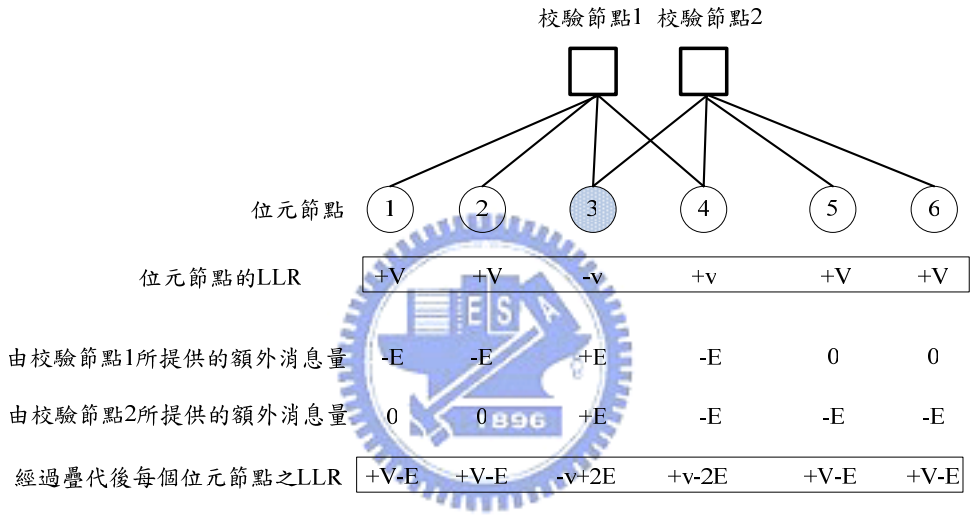


圖 4-3 高密度位置發生一個錯誤 (位元節點 3) 時的訊息傳遞情況

若是利用 JN 演算法同樣考慮發生一個錯誤的情況，我們首先可以利用前一章中所提到的高斯消去動作，得到簡化梯式校驗矩陣和相對應的 Tanner 圖。這個動作能夠使得低信任度的位元節點僅連接到一個校驗節點，每個校驗節點也僅連結至一個低信任度的位元節點，因此校驗矩陣高密度的位置此時會均為高信任度之位元節點。利用這個動作，我們將上例中低信任度的位元節點 3 和 4 移至低密度位置，而位元節點 1 和 6 轉移至高密度位置，如圖 4-4 所示。再利用此 Tanner 圖計算每個位元節點的額外消息量，圖 4-5 所示。圖 4-5 中可以發現，相較於圖 4-3，錯誤傳遞的情況大量減少，僅有一個高信任度位置會收到錯誤的額外消息量。

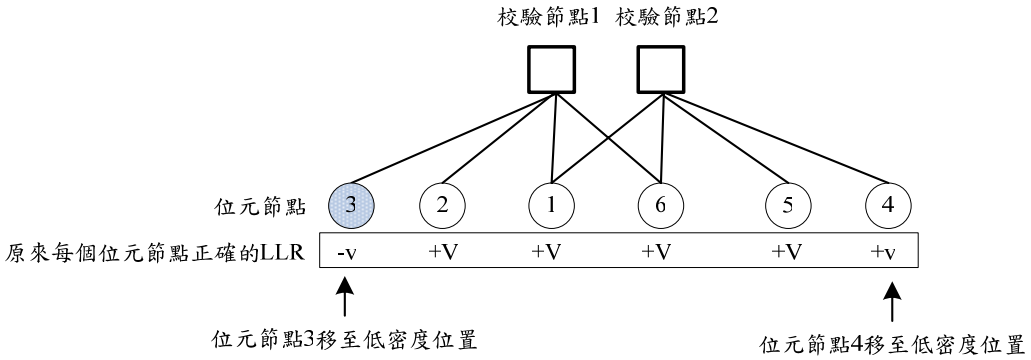


圖 4-4 經過 JN 演算法高斯消去動作後的 Tanner 圖

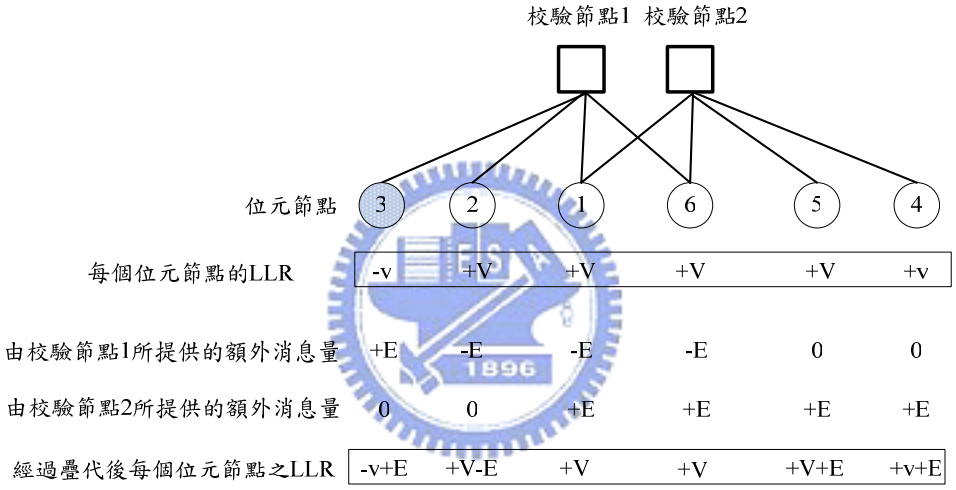


圖 4-5 發生一個錯誤 (位元節點 3)時，經高斯消去動作後的額外消息量傳遞情形

接下來考慮兩個低信任度位置 (位元節點 3 和位元節點 4) 同時發生錯誤的情況，我們依照前例中所做的假設，給予位元節點 3 和位元節點 4 為-v 的 LLR 值，如圖 4-6 所示。而此時的訊息傳遞情形表示於圖 4-7。從圖 4-7 中我們可以發現，由於錯誤間互相影響，本來錯誤的位置不斷收到錯誤的額外消息量，這個現象在解碼上亦容易造成解碼失敗。

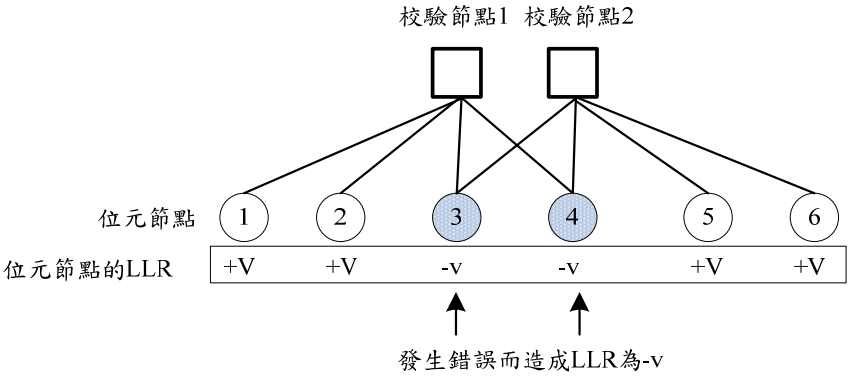


圖 4-6 當高密度位置發生兩個錯誤時 (位元節點 3、位元節點 4)

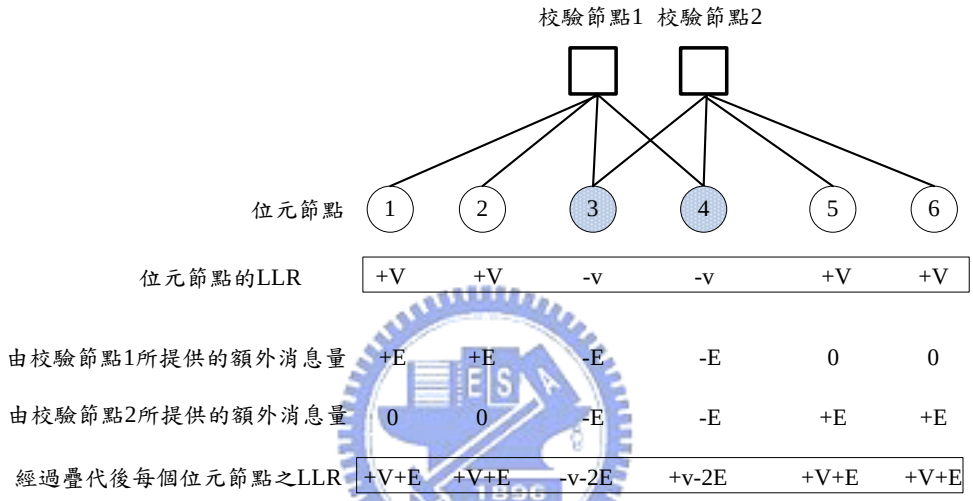


圖 4-7 當高密度位置發生兩個錯誤時，額外消息量的傳遞情形

若在同樣的情況下，使用 JN 演算法中高斯消去法的動作，使得低信任度位置自高密度位置改至低密度位置，則經此動作後的 Tanner 圖和訊息傳遞情況可以圖 4-8 和圖 4-9 呈現。由圖 4-9 可以發現，在此例中雖然也出現了大量的錯誤傳遞情形，但若在信任度 V 夠大的情況下，在此次疊代過程中較能不被錯誤的額外消息量影響而改變其正負號，並且低信任度的位置有機會在這次疊代中被更正，相較於圖 4-7 的情況，高斯消去動作仍然較能抑制解碼失敗的發生。

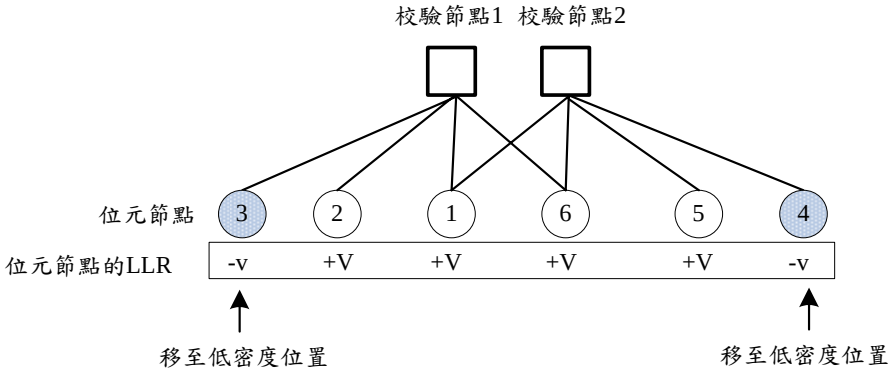


圖 4-8 兩個錯誤下，經 JN 演算法高斯消去動作後的 Tanner 圖

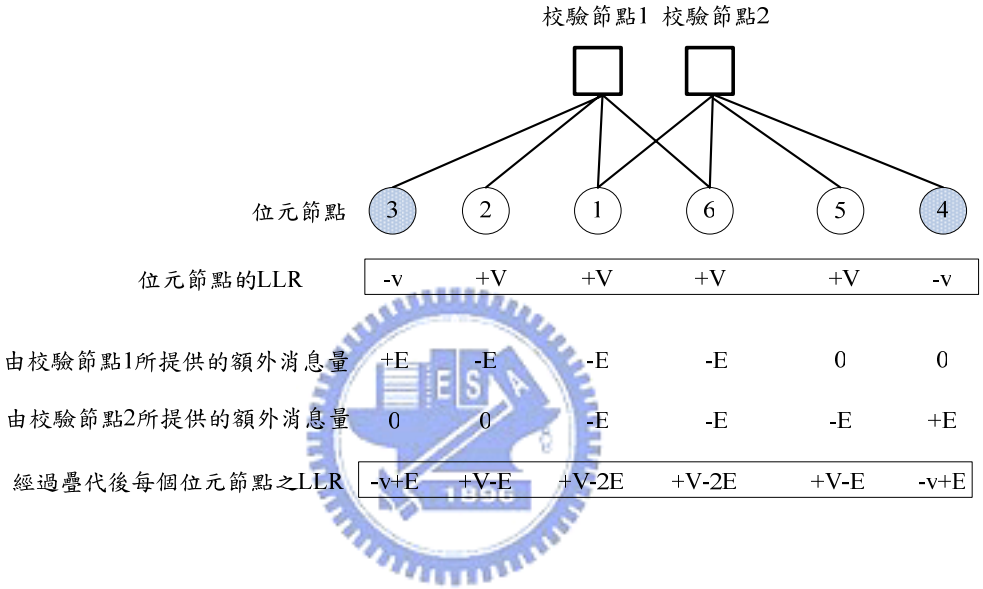


圖 4-9 發生兩個錯誤時，經高斯消去法作用後的訊息傳遞情形

由以上的分析可知，JN 演算法的高斯消去動作能夠抑制低信任度錯誤所造成的問題，此為其優勢。

4-1.2 JN 演算法之潛在問題

在前一小節的討論中，雖然 JN 演算法利用高斯消去動作將低信任度的位元節點置於二位元奇偶校驗矩陣的低密度位置，使其僅影響到一個校驗節點以減少錯誤傳遞的情況，然而這個動作建立在相信高信任度位置均為正確的假設之下，實際上卻並非如此。因此 JN 演算法的潛在問題便在於過於相信高信任度位置為正確的假設，而將其置於二位元奇偶校驗矩陣的高密度位置，若高信任度位置發生錯誤的情況，僅管錯誤的個數不多，卻往往會造成嚴重的錯誤傳遞現象，使得

解碼器收斂至錯誤的碼字，甚至無法收斂。

圖 4-10 為統計上 JN 演算法於(15,7)里德所羅門碼解碼失敗分布圖，橫軸代表若無法成功解碼時，於第一次高斯消去動作執行時，被置於高密度位置的錯誤位元節點個數，而縱軸代表所有測試碼字中，發生此個數情況的次數，此時所使用的最大疊代次數$l_{\max}$為 20 次，α 為 0.1。

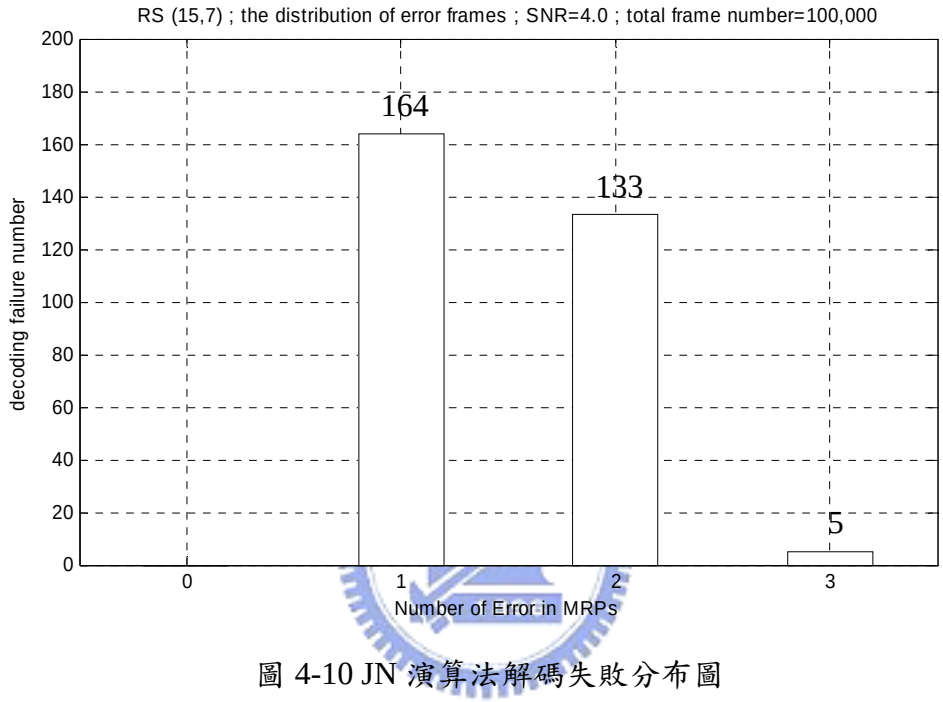


圖 4-10 JN 演算法解碼失敗分布圖

由圖 4-10 可以觀察到，若於第一次的高斯消去動作中，錯誤的發生均落於低密度位置 (即橫軸坐標為 0)時，則均可順利解碼成功。由前一小節的討論可知，JN 演算法於此時提供了良好的保護能力。但當第一次的高斯消去動作中高密度位置的錯誤個數增加時，JN 演算法卻由於過度相信這些高密度位置為正確的位置，造成解碼失敗的結果。於下一小節中，我們將提出適當的改善機制來避免這個問題。

4-2 提出之演算法

4-2.1 提出之演算法概念說明

當 JN 演算法執行時，其所需要擔心的問題為錯誤傳遞的現象。若要抑制錯

誤傳遞的現象，則需要於疊代過程中即進行處理，否則當碼字已逐漸收斂到錯誤的碼字時，再進行補救則為時已晚。然而 JN 演算法本身的特性在於相信錯誤均落於 LRIPs 上而進行解碼，因此若要幫助 JN 演算法解決前一小節中所提到的問題，我們傾向於利用額外的機制來提供 JN 演算法足夠的訊息量來進行改善。這些訊息量能夠快速使得 LRIPs 的錯誤被更正，並且加強其信任度後，使得 MRPs 和 LRIPs 進行交換。此時 MRPs 若發生錯誤，有可能在幾次疊代後即落入為 LRIPs，而使得解碼過程得以不受到原來 MRPs 錯誤的影響而收斂到正確的碼字。

於本論文中希望使用額外的機制為 OSD(1) 解碼演算法。該演算法能夠利用測試錯誤樣本 (TEP) 和 MRIPs 進行重新編碼的動作，再自這些重新編碼的碼字中選擇和接收信號相關性較大的碼字。這些和接收信號相關性較大的碼字，其於 LRPs 的正確性也會和正確的碼字較為接近。因此若以 OSD(1)的解碼結果提供適當的消息量給予 JN 演算法的 LRIPs，則能夠協助 JN 演算法解決前述的潛在問題。

在此說明提出之演算法和 JN-OSD(1)解碼演算法之差別在於，JN-OSD(1)並未提供適當的機制來解決 JN 演算法面對高信任度錯誤的問題，僅管 OSD(1) 本身對於碼字中能夠提供高信任度位置中一個錯誤位置的更正，然而在多個高信任度錯誤發生的情況下，其並不能協助 JN 演算法進行妥善的處理。

因此本論文提出之演算法其能夠提供 JN 演算法以下的兩項協助：

(1) 加速 LRIPs 錯誤位置之更正，和 LRIPs 正確位置之信任度提昇。

(2) 藉由使 MRPs 和 LRIPs 之交換，抑制 MRPs 錯誤的影響。

提出之演算法的解碼疊代過程可以表示如圖 4-11 所示。

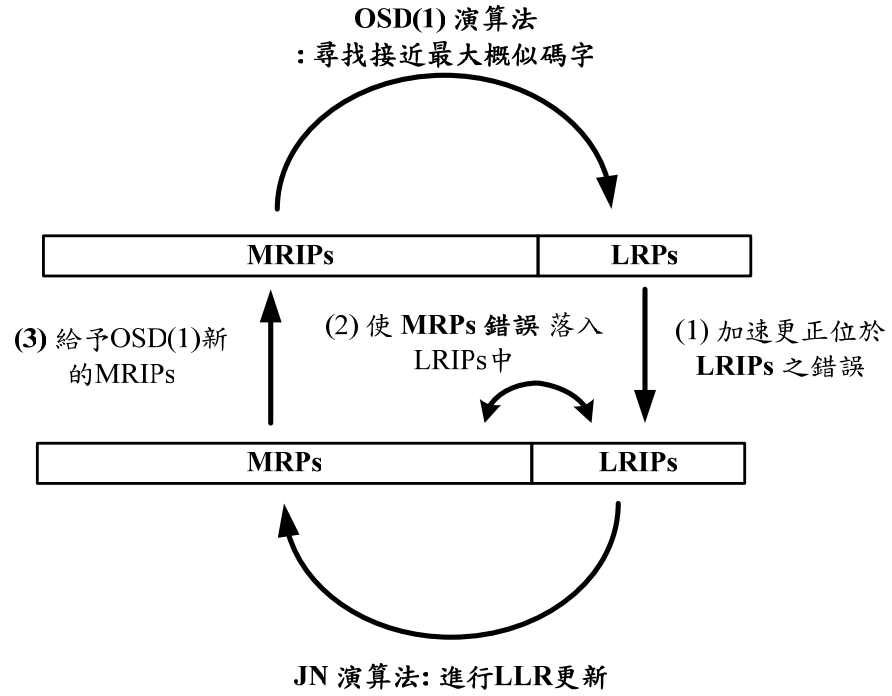


圖 4-11 所提供演算法之概念圖

4-2.2 提出之演算法解碼步驟

以下將說明於(n,k)線性區塊碼時，提出之演算法的解碼步驟：

步驟一、設定抑制係數 α， 最大疊代次數$l_{\max}$，並計算每個位元節點的初始的 LLR 為$L^{(0)}(c_i)=(4y_i/N_0)$，並以初始 LLR ($L^{(0)}(c_i)$) 開始進行解碼。

步驟二、OSD(1)演算法階段：

(1) 依據 LLR 絕對值$\left|\underline{L}^{(l)}\right|$的大小將每個位元位置進行排序，並且記錄排序後的順序。

(2) 依排序的結果，依信任度由大至小的位置對二位元生成矩陣$\mathbf{G}_b$進行高斯消之動作而得到簡化梯式生成矩陣。若該位置無法經列運算正規化，則跳過該位置至信任度次大的下一個位置，直到得到k個 MRIPs 後停止。

(3) 此時假設此k個 MRIPs 中僅有一個以下的錯誤。因此生成 1 的個

數為 0，和 1 的個數為 1 的 TEPs。接下來按照此$k+1$個 TEPs 以高斯消去動作後的生成矩陣進行編碼得到$k+1$組候選碼字(candidate codewords)。

(4) 若自候選碼字中找到最大概似碼字，則停止解碼並輸出結果。若到達最大預定疊代次數$l_{\max}$時，則停止解碼並輸出最目前所存相關性最大的碼字。否則保留此次疊代中和接收信號相關性最大的碼字，並進行步驟三。

步驟三、 回授機制：回授機制將說明於 4-2.3 節中。

步驟四、 JN 演算法階段。

(1) 更新奇偶校驗矩陣：$\mathbf{H}_b^{(l)} = \phi\left(\mathbf{H}_b, \left|\underline{L}^{(l)}\right|\right)$，$\phi$表示更新$\mathbf{H}_b$的動作。依據回授機制後的 LLR 絕對值$\left|\underline{L}^{(l)}\right|$的大小將每個位元位置進行排序，並且記錄排序後的順序。利用高斯消去法，按照信任度由小至大的位置將$\mathbf{H}_b^{(l)}$消為簡化梯式矩陣，並得到$(n-k)$個 LRIPs。

(2) 利用高斯消去法正規化後的$\mathbf{H}_b^{(l)}$，進行梯度下降法得到每個位元節點的額外消息量：$\underline{L}_{ext}^{(l)} = \varphi\left(\mathbf{H}_b^{(l)}, \underline{L}^{(l)}\right)$，即式 (2.14) 的過程。$\varphi$表示計算額外消息量之動作。

(3) 更新每個位元節點的 LLR： $\underline{L}^{(l+1)} = \underline{L}^{(l)} + \alpha \underline{L}_{ext}^{(l)}$，即式 (2.13) 的過程。在這裡$0 < \alpha \le 1$。

(4) 回到步驟二。

4-2.3 提出演算法之 OSD(1) 回授機制討論

於回授機制的討論中，本論文使用四種方式進行模擬以選擇回授機制：

回授機制(1)：

於每次疊代過程中，儲存 OSD(1)機制中所得到與接收信號相關性最大的碼

字。若於 OSD(1)機制結束後，該碼字於某 LRP 的位置判斷為 0，則於該位置的 LLR 加上 $+A$ 的額外消息量，若該碼字於某 LRP 的位置判斷為 1，則於該位置的 LLR 加上 $-A$ 的額外消息量，其中 A 為可調整之係數，且 $A>0$ 。

回授機制(2)：

於每次疊代過程中，儲存 OSD(1)機制中所得到與接收信號相關性最大的兩個碼字。若於 OSD(1)機制結束後，該兩個碼字於某 LRP 的位置判斷同為 0，則於該位置的 LLR 加上 $+A$ 的額外消息量，若該兩個碼字於某 LRP 的位置判斷為同為 1，則於該位置的 LLR 加上 $-A$ 的額外消息量，其中 A 為可調整之係數，且 $A>0$ 。

若此兩個碼字的判斷不相同時，則分別計算該兩個碼字的事後機率，將該位置為判斷為 0 的碼字事後機率(p_0)置於分子，而該位置判斷為 1 的碼字事後機率置於分母(p_1)，最後將該位置原有的 LLR 加上 $\log\left(p_0/p_1\right)$ 的額外消息量，若 $\log\left(p_0/p_1\right)$ 超過 $+A$ 或小於 $-A$ 時，該位置原有的 LLR 僅加上 $+A$ 或 $-A$ 之值。

回授機制(3)：

首先於每次疊代過程中，首先計算 OSD(1) 所有生成碼字的事後機率。我們定義於生成碼字中第 i 個位置為 0 的碼字集合為 C_{0i} ，而生成碼字中第 i 個位置為 1 的碼字集合為 C_{1i} 。則接下來我們將 LRPs 中位元節點 i 原有的 LLR 加上 $\log\left(\Pr\left(\underline{c}_{0\max}\mid\underline{r}\right)/\Pr\left(\underline{c}_{1\max}\mid\underline{r}\right)\right)$ 的額外消息量值，其中 $\underline{c}_{0\max}$ 為 C_{0i} 中事後機率最大的碼字，$\underline{c}_{1\max}$ 為 C_{1i} 中事後機率最大的碼字。若 $\log\left(\Pr\left(\underline{c}_{0\max}\mid\underline{r}\right)/\Pr\left(\underline{c}_{1\max}\mid\underline{r}\right)\right)$ 超過 $+A$ 或小於 $-A$ 時，該位置原有的 LLR 僅加上 $+A$ 或 $-A$ 之值。

回授機制(4)：

於每次疊代過程中，首先計算 OSD(1) 所有生成碼字的事後機率。我們定義於生成碼字中第 i 個位置為 0 的碼字集合為 C_{0i} ，而生成碼字中第 i 個位置為 1 的碼字集合為 C_{1i} 。則接下來我們將 LRPs 中位元節點 i 原有的 LLR 加上

$\log\left(\sum_{\underline{c}\in C_{0i}}\Pr(\underline{c}\,|\,\underline{r})/\sum_{\underline{c}\in C_{1i}}\Pr(\underline{c}\,|\,\underline{r})\right)$ 的額外消息量值。若 $\log\left(\sum_{\underline{c}\in C_{0i}}\Pr(\underline{c}\,|\,\underline{r})/\sum_{\underline{c}\in C_{1i}}\Pr(\underline{c}\,|\,\underline{r})\right)$ 超過 $+A$ 或小於 $-A$ 時，該位置原有的 LLR 僅加上 $+A$ 或 $-A$ 之值。

接下來利用 (15,7) 里德所羅門碼對於以上四種回授機制進行模擬，抑制參數 $\alpha=0.10$，最大疊代次數為 20 次，在不同的 A 值情況下，模擬結果分別於圖 4-12 至圖 4-17。

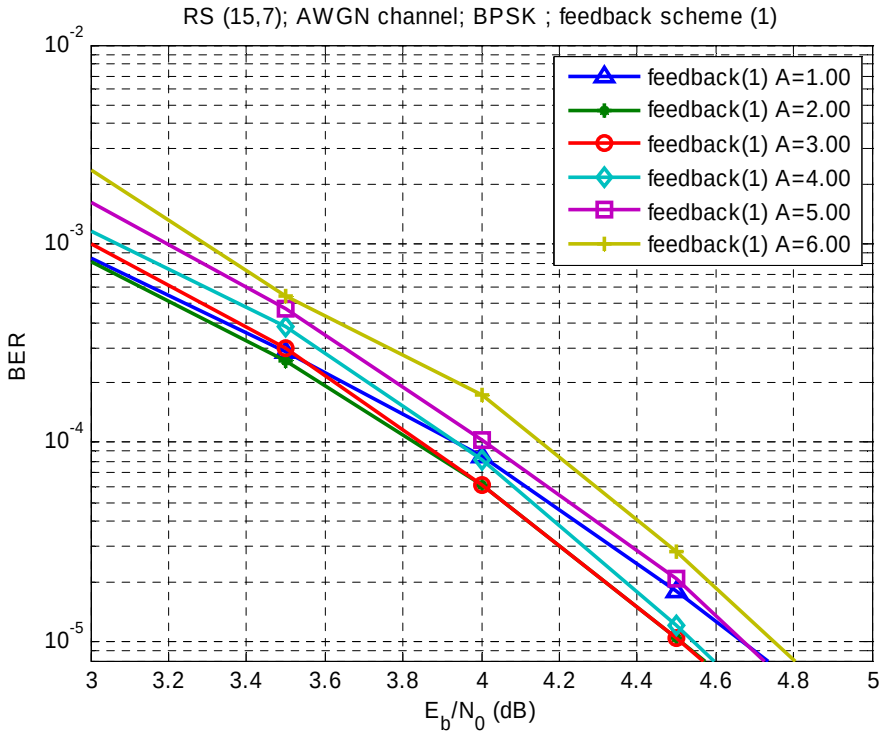


圖 4-12 (15,7)里德所羅門碼配合回授機制(1)於不同 A 值下的情況

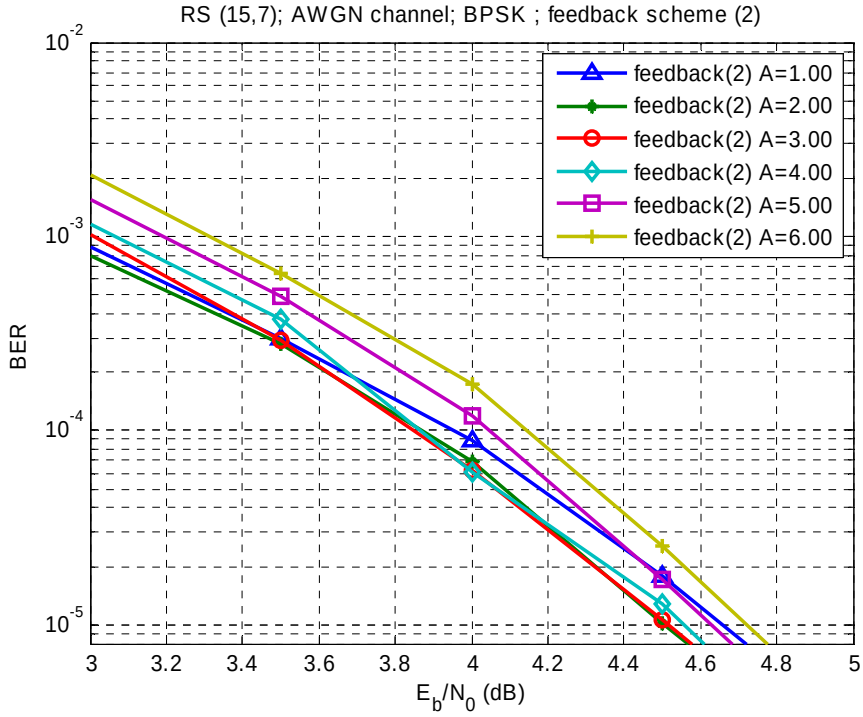


圖 4-13 (15,7)里德所羅門碼配合回授機制(2)於不同 A 值下的情況

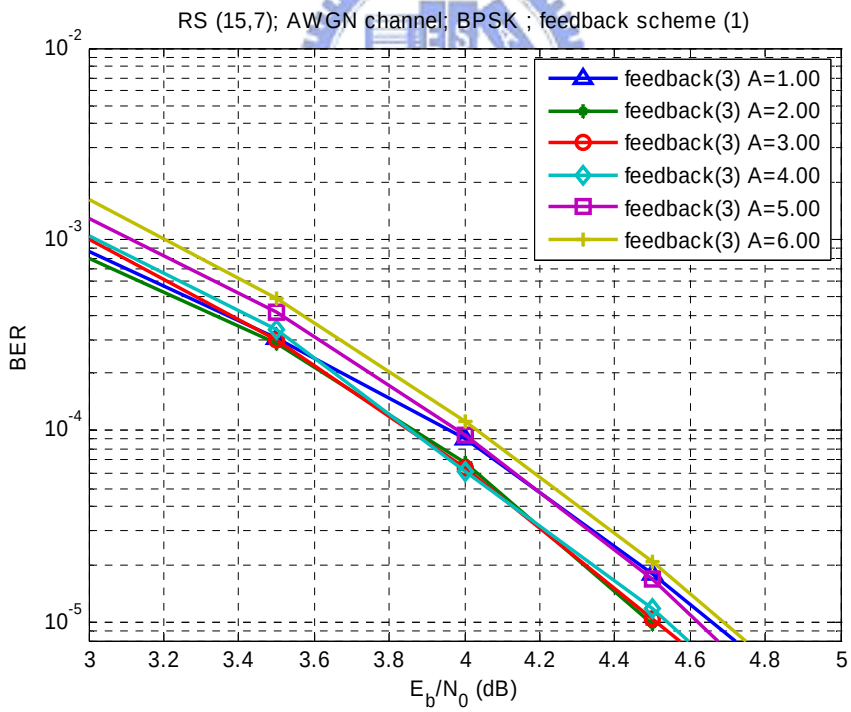


圖 4-14 (15,7)里德所羅門碼配合回授機制(3)於不同 A 值下的情況

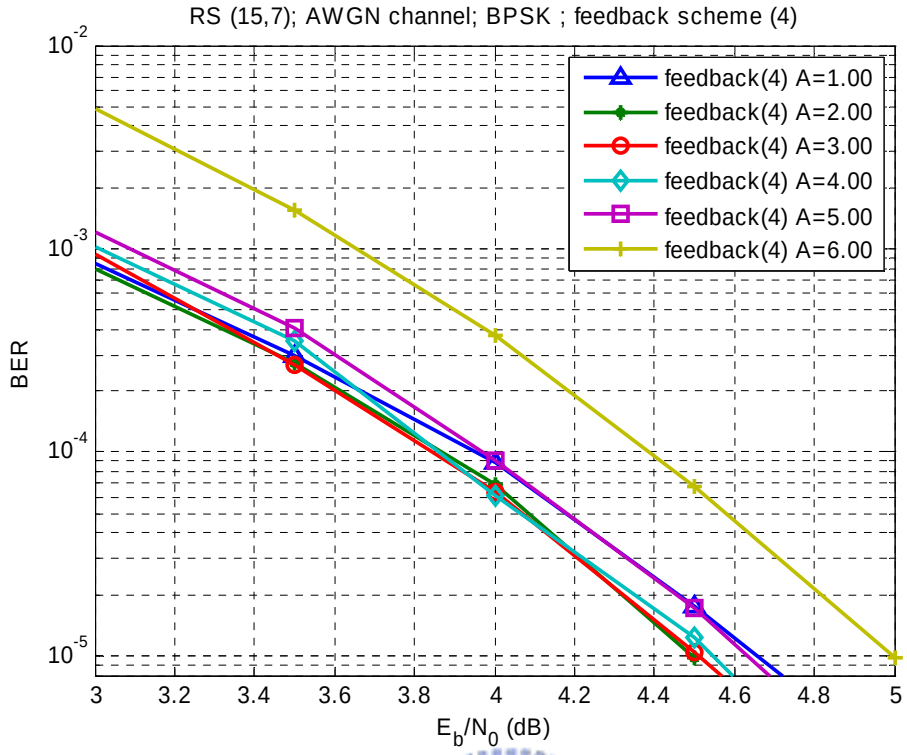


圖 4-16 (15,7)里德所羅門碼配合回授機制(3)於不同 A 值下的情況

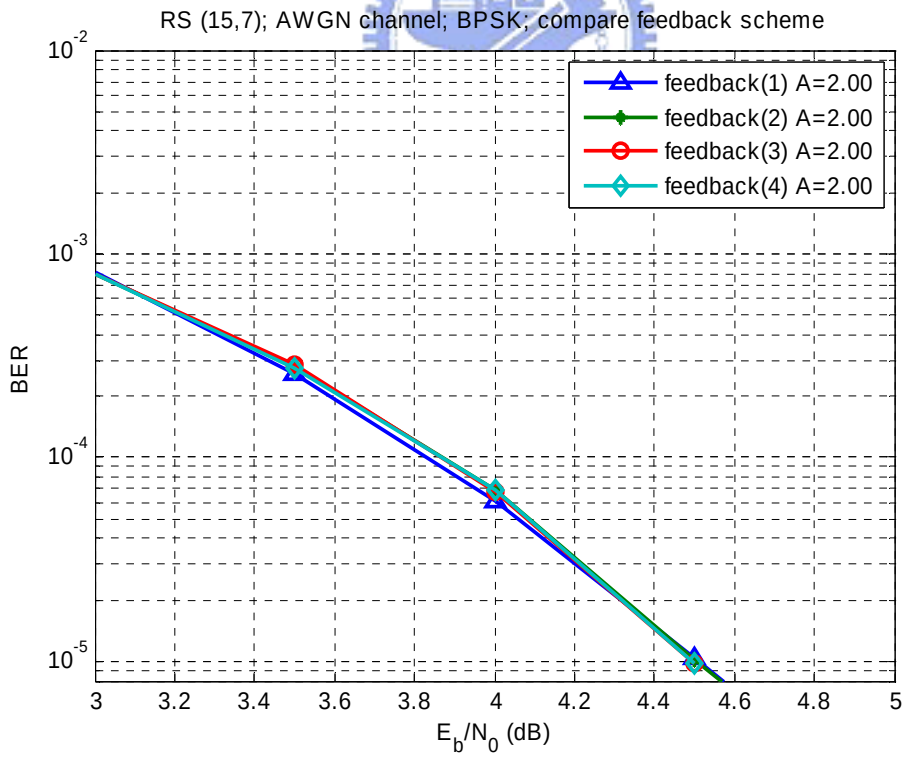


圖 4-17 (15,7)里德所羅門碼比較不同回授機制於最好 A 值下的情況

於圖 4-17 的比較中可以了解，此四種回授機制在解決 JN 演算法潛在問題的作用上差別並不大。因此基於簡化運算的考量，本論文最終選擇回授機制(1) 應用於接下來的討論中。所提出演算法之相關效能模擬結果將於下一章中呈現並進行討論。

4-2.4 提出之演算法數據觀察

接下來由模擬數據來觀察並說明提出之演算法對於 JN 演算法解碼過程所造成的影響。以下為(15,7) 里德所羅門碼使用提出之演算法，和原有的 JN-OSD(1) 演算法相比較，測試其同時在面對多個高信任度位置錯誤時的位元節點 LLR 隨疊代次數的變化情形。此時提出之演算法的抑制參數$\alpha = 0.10$，最大疊代次數為 20 次，A 值為 2.00。JN-OSD(1)演算法之參數$\alpha = 0.10$，最大疊代次數為 20 次。

在以下的模擬中，初始錯誤發生於位元節點 1、8、16、19、29、42、53、56 的位置，其中位元節點 8 和位元節點 19 於一開始被 JN 演算法列為高密度的位置，其餘錯誤則被列為低密度位置。這些錯誤位置的 LLR 的變化分別表示於圖 4-18 至圖 4-26 中，橫座標表示疊代次數，縱座標則表示每個位元節點的 LLR。對於提出之演算法，由於每次疊代可分為 OSD(1)回授結構所給予的額外消息量，和 JN 演算法所帶來的額外消息量。因此所提出演算法在圖 4-18 至圖 4-26 中，每一疊代間隔的前半部代表 OSD(1)回授結構所造成的 LLR 變化，而後半部代表 JN 演算法所帶來的影響。對於 JN-OSD(1)演算法的圖形，每一間隔代表的僅有 JN 演算法所帶來的影響。

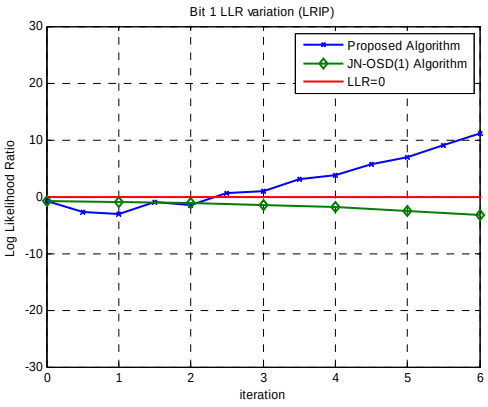


圖 4-18 位元節點 1 的 LLR 變化

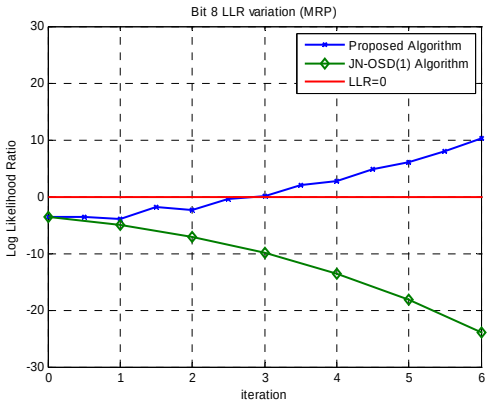


圖 4-19 位元節點 8 的 LLR 變化

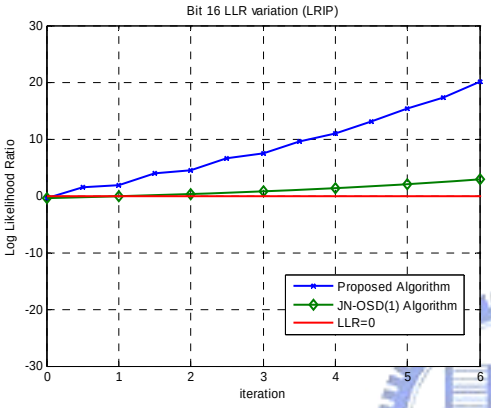


圖 4-20 位元節點 16 的 LLR 變化

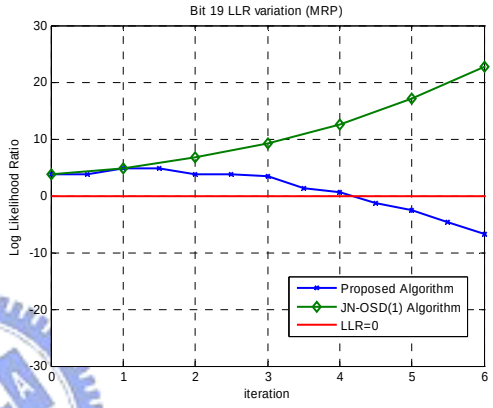


圖 4-21 位元節點 19 的 LLR 變化

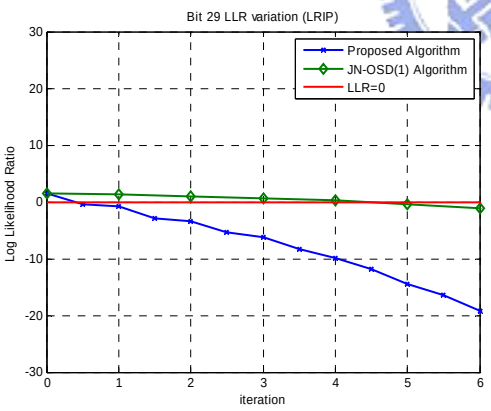


圖 4-18 位元節點 29 的 LLR 變化

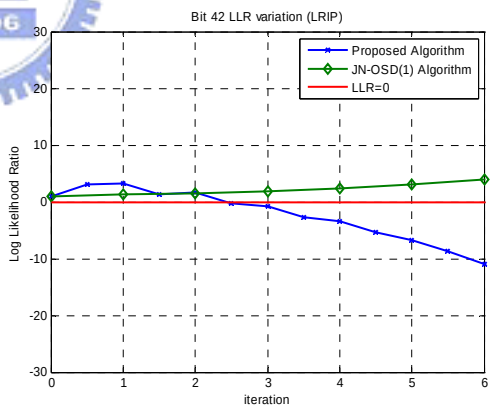


圖 4-19 位元節點 42 的 LLR 變化

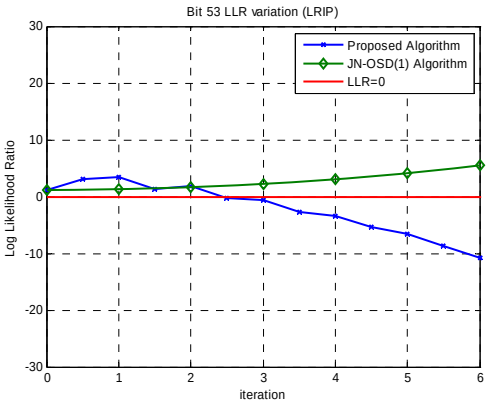


圖 4-22 位元節點 53 的 LLR 變化

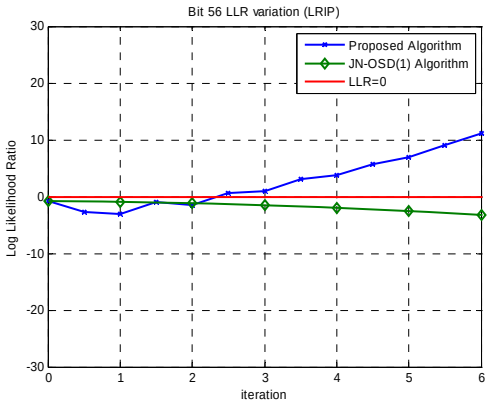


圖 4-23 位元節點 56 的 LLR 變化

由以上數據可以觀察到，這些錯誤位置於 JN-OSD(1)演算法大部分無法被更正，但在提出之演算法的運作下均順利被更正了。因此提出之演算法的確抑制了高信任度錯誤位置對於 JN 演算法的影響，提供了加速 LRIPs 錯誤位置之更正、LRIPs 正確位置之信任度提昇等優點，並藉由使 MRPs 和 LRIPs 之交換，抑制 MRPs 錯誤的影響，進而改善原有 JN-OSD(1) 解碼結構的潛在問題。

於圖 4-24 中，列出提出之演算法和 JN-OSD(1)解碼演算法、JN 演算法對於高信任度位置造成(15,7)里德所羅門碼解碼失敗之分布圖。同樣地，橫軸代表若無法成功解碼時，於第一次高斯消去動作執行時，被置於高密度位置的錯誤位元節點個數，而縱軸代表所有測試碼字中，發生此個數情況的次數。此時 JN 演算法、JN-OSD(1)演算法和提出之演算法的 $\alpha = 0.10$，最大疊代次數為 20 次，提出之演算法的 A 值則為 2.00。

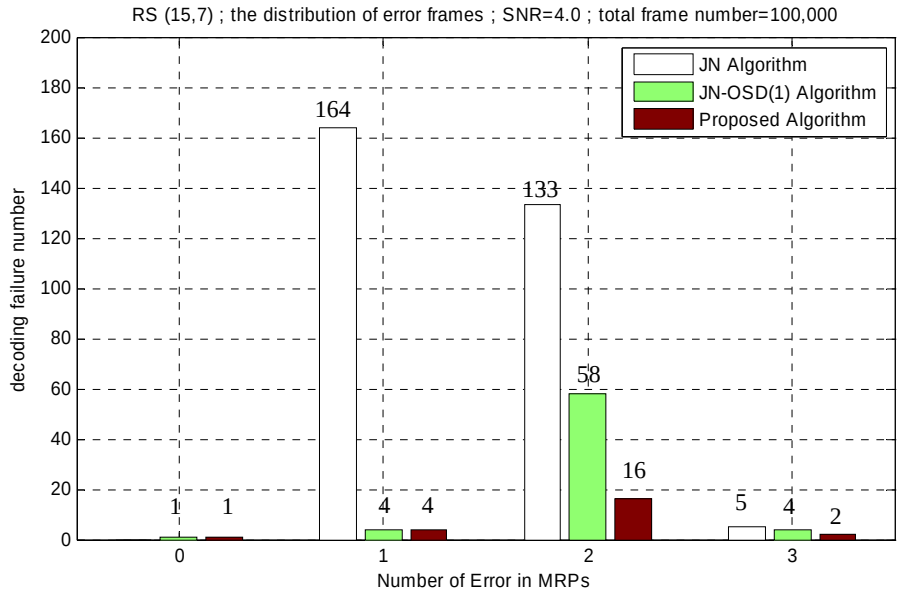


圖 4-24 三種演算法之解碼失敗情況之比較圖

由圖 4-24 中可以發現，提出演算法的確減少了由於高信任度位置造成的解碼失敗情況。有時候，錯誤均發生在 LRIPs 的情況下，提出之演算法或 JN-OSD(1)演算法中，所能找到和接收信號相關性最大的候選碼字不是正確的碼字，反而 JN 演算法在此情況下才能夠順利抑制所有的 LRIPs 錯誤而找到正確的碼字。因此才會出現 MRPs 錯誤是 0 個時，提出之演算法或 JN-OSD(1)演算法反而無法成功解碼的特殊情況，但此情況相當少。

4-3 結合里德所羅門碼之內插特性之方式

雖然所提出演算法已使得 JN 演算法因高信任度位置錯誤造成解碼失敗的情況受到改善，然而仍然有未能被更正的錯誤情況。因此在此提出利用里德所羅門碼未知位置解碼器 (erasure decoder)，進行隨機內部疊代內插出候選的碼字之概念以輔助提出之演算法，此概念整體運作如圖 4-25 所示。

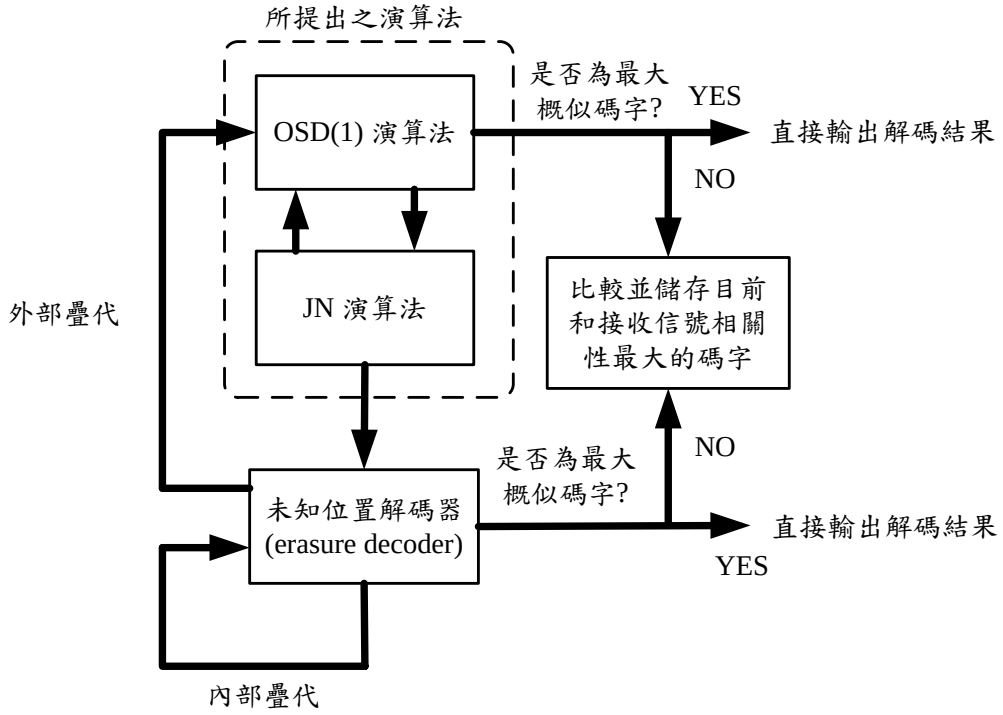


圖 4-25 所提出演算法結合里德所羅門碼之未知位置解碼器之運作方式。

里德所羅門碼內插器內部疊代過程如下：

步驟一、 將經過 JN 演算法之 LLR 進行硬式決策後，將二位元碼字轉為符元碼字。

步驟二、 自符元碼字中隨機選取 N-K 個位置後，將這些位置自碼字中移去。

步驟三、 由剩下的 K 個位置進行內插出所移去的 N-K 個位置 [17]：

(1) 由移去的 N-K 個符元位置計算未知位置多項式 (erasure location polynomial)：

$$\tau(x)=\prod_{l=1}^{N-K}\left(1+y_{i_l}x\right) \tag{4.1}$$

其中 $y_{i_l}=\beta^{i_l}$，其表示未知位置的所在於符元碼字中第 i_l 個位置。

(2) 計算修正 Forney 徵狀 (Forney syndrome) $T(x)$，其中 t 為該里德所羅門碼之更正能力:

$$T(x)=S(x)\tau(x)\bmod x^{2t+1} \tag{4.2}$$

其中 $S(x)$ 為徵狀方程式 (syndrome polynomial)，計算方式為：

$$S(x)=1+S_1x+S_2x^2+S_3x^3+\cdots+S_{N-K}x^{N-K} \tag{4.3}$$

$$S_i=r(\alpha^i)=r_1+r_2(\alpha^i)+r_2(\alpha^i)^2\cdots+r_N(\alpha^i)^{N-1} \tag{4.4}$$

$r(x)$ 為符元碼字之碼字多項式，其中 r_j 表示碼字中第 j 個位置之符元，$1\le j\le N$ 。

(3) 此 N-K 個位置經過內插後所得的值為

$$f_{i_l}=\frac{y_{i_l}\bullet T\left(y_{i_l}^{-1}\right)}{\tau'\left(y_{i_l}^{-1}\right)}\text{，}\forall 1\le l\le N-K \tag{4.5}$$

步驟四、 驗證所內插出的碼字為一合法碼字後，將符元碼字轉為二位元碼字，若此二位元碼字為最大概似碼字，則輸出結果並停止解碼，否則將該二位元碼字和目前與接收信號相關性最大的二位元碼字進行比較，若該二位元碼字相關性較大，則保留該碼字，並進行步驟五。

步驟五、 若未達最大內部疊代次數時，則回到步驟二進行下一次內部疊代。若已達最大內部疊代次數，則回到 OSD(1)進行下一次外部疊代。

利用此種方式在適當調整內部疊代次數後，能夠有機會內插出相關性較大的碼字。效能增益配合提出之演算法於 (15,7) 的里德所羅門碼的模擬結果於圖 4-26，此時之 $\alpha=0.1$ ， $l_{\max}=20$ ， $A=2.0$ 。圖中 Inner round 所代表的是所需的內部疊代次數，分別為 50 次、70 次和 100 次，其約能較所提出演算法增加 0.1dB 之效能。然而由於其所要付出的複雜度極高，因此僅將此概念在此說明。

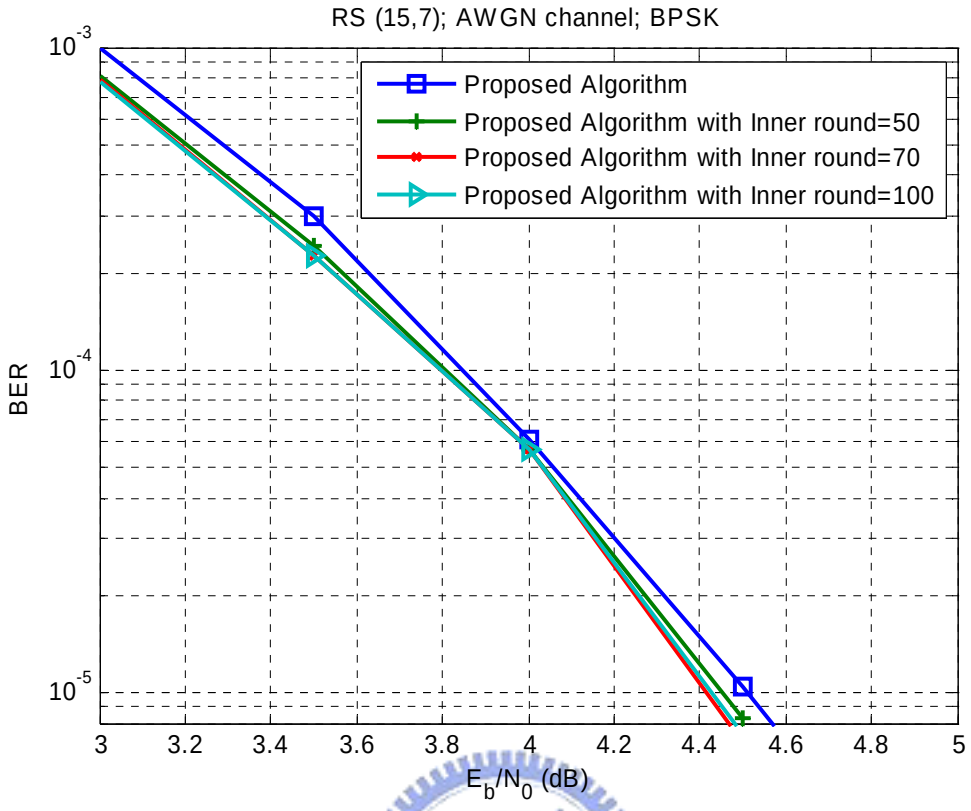


圖 4-26　(15,7)里德所羅門碼利用未知位置解碼器在不同內部疊代時的效能改善

第五章

模擬結果與討論

在本章中，我們將提出之演算法於里德所羅門碼的模擬結果於此呈現，並且對模擬結果進行討論。以下的提出之演算法運作中，不包含前一章最後所提的里德所羅門碼內插機制。

首先於提出之演算法中，選擇的 A 值將會影響提出之演算法效能，因此我們在以下的模擬中，首先調整 A 值以達到最佳的效能，以下的模擬中，最佳的 A 均為 2.0。圖 5-1 至圖 5-5 為不同的里德所羅門碼在不同 A 值時的效能表現。以下所提出演算法的模擬中， (15,7) 里德所羅門碼的最大疊代次數為 20 次，抑制係數 α 為 0.1。(15,9) 里德所羅門碼的最大疊代次數為 20 次，抑制係數 α 為 0.1。(31,15) 里德所羅門碼的最大疊代次數為 20 次，抑制係數 α 為 0.05。(31,23) 里德所羅門碼的最大疊代次數為 20 次，抑制係數 α 為 0.1。(31,25) 里德所羅門碼的最大疊代次數為 20 次，抑制係數 α 為 0.05。

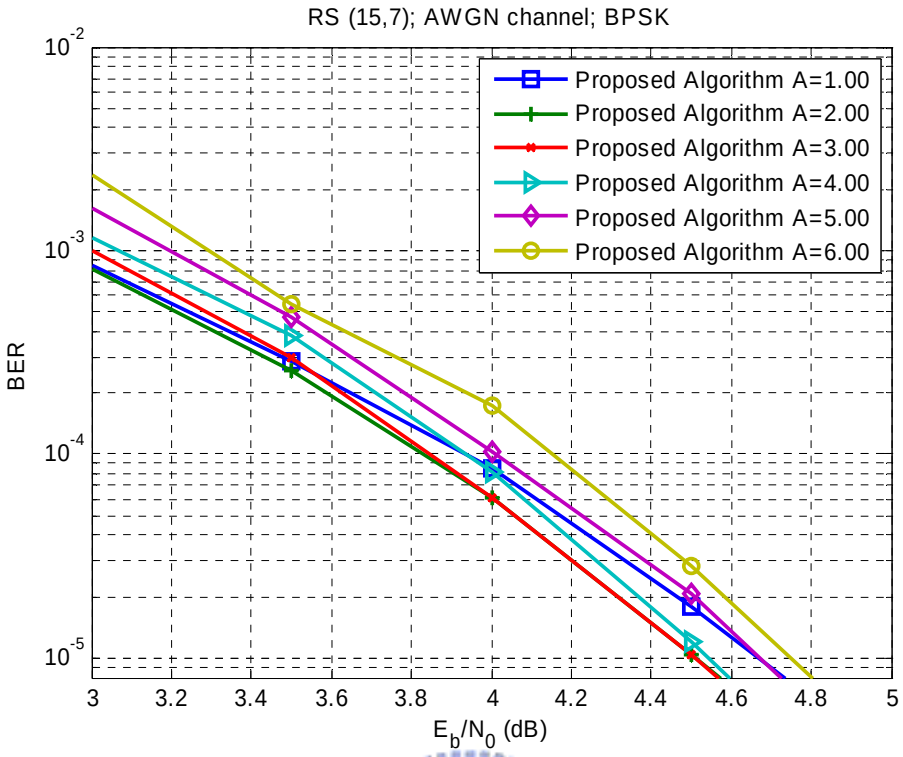


圖 5-1　(15,7) 里德所羅門碼於提出之演算法在不同 A 值時的表現

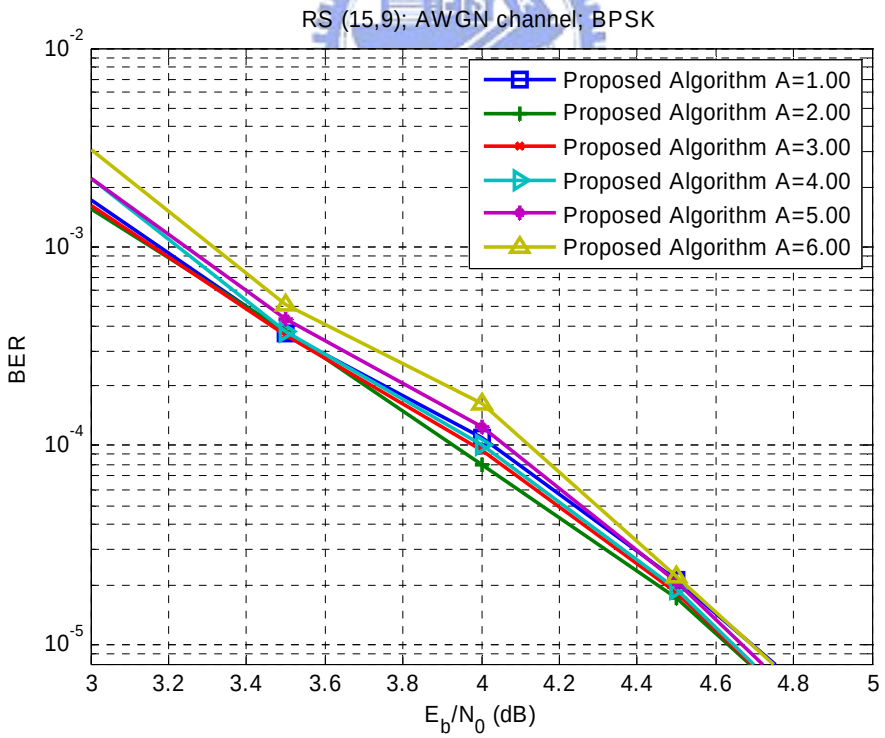


圖 5-2　(15,9) 里德所羅門碼於提出之演算法在不同 A 值時的表現

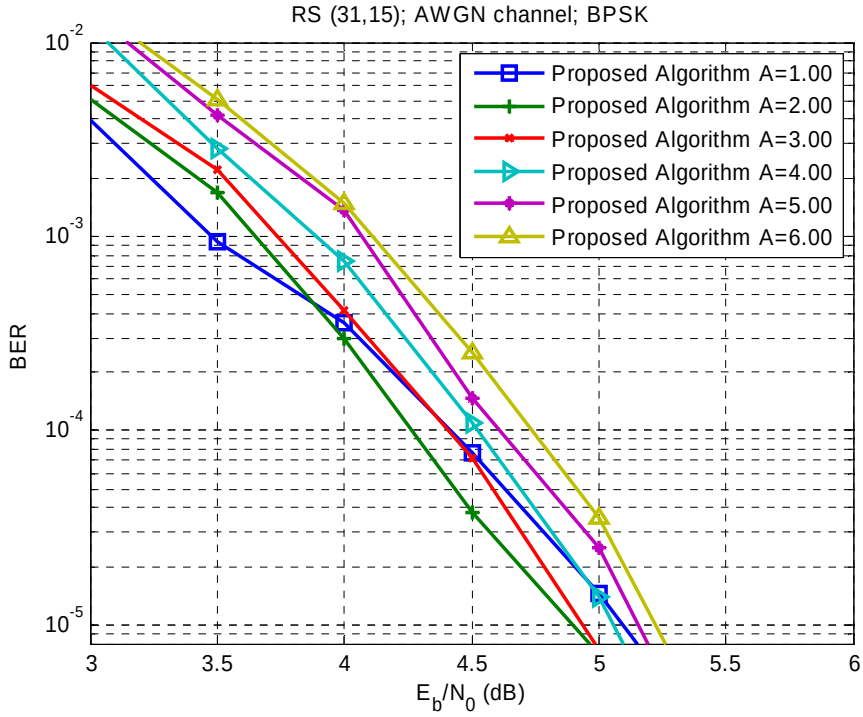


圖 5-3　(31,15) 里德所羅門碼於提出之演算法在不同 A 值時的表現

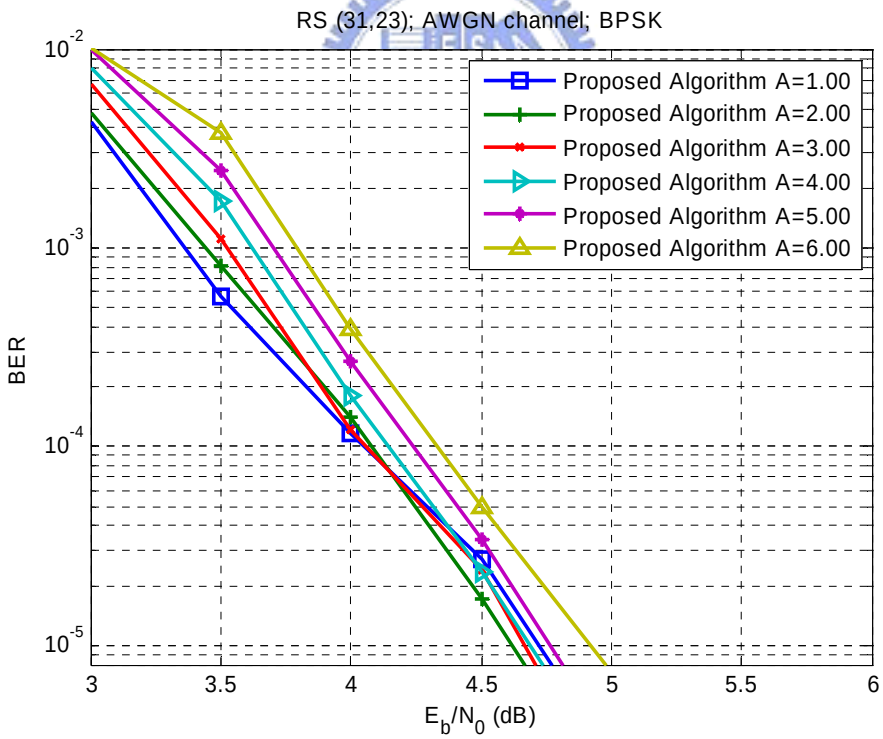


圖 5-4　(31,23) 里德所羅門碼於提出之演算法在不同 A 值時的表現

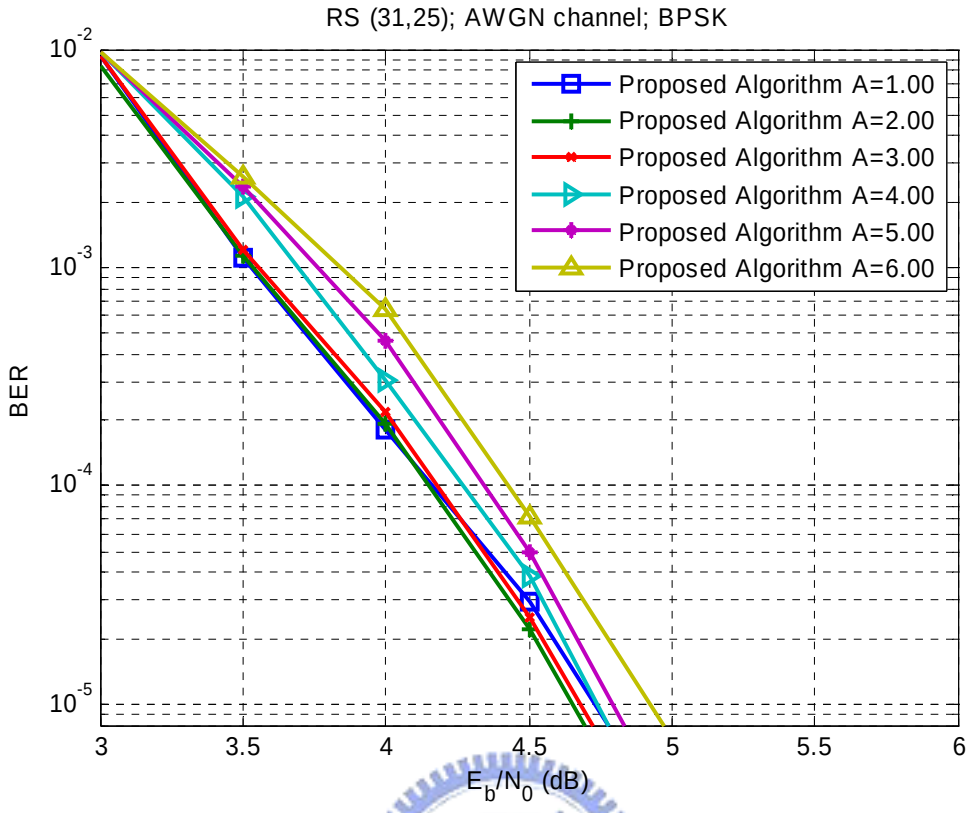


圖 5-5　(31,25) 里德所羅門碼於提出之演算法在不同 A 值時的表現

而從模擬結果選擇出最佳的 A 值後，我們將提出演算法和其他演算法間的效能列出於圖 5-6 至圖 5-10 中。

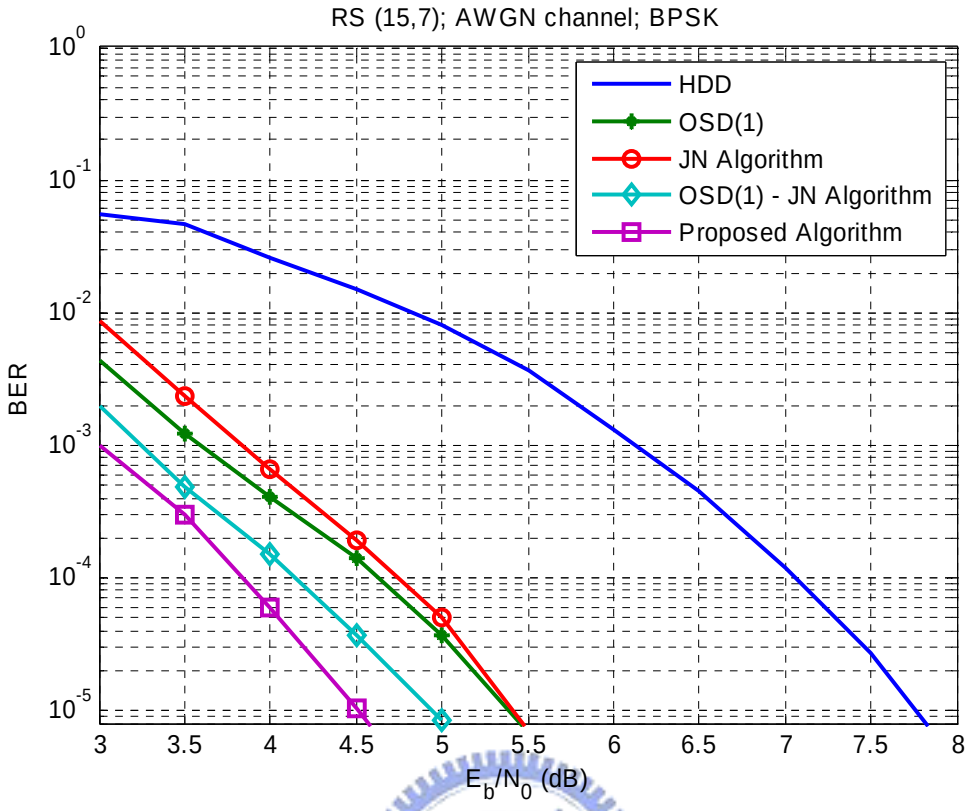


圖 5-6 (15,7) 里德所羅門碼在不同演算法間表現

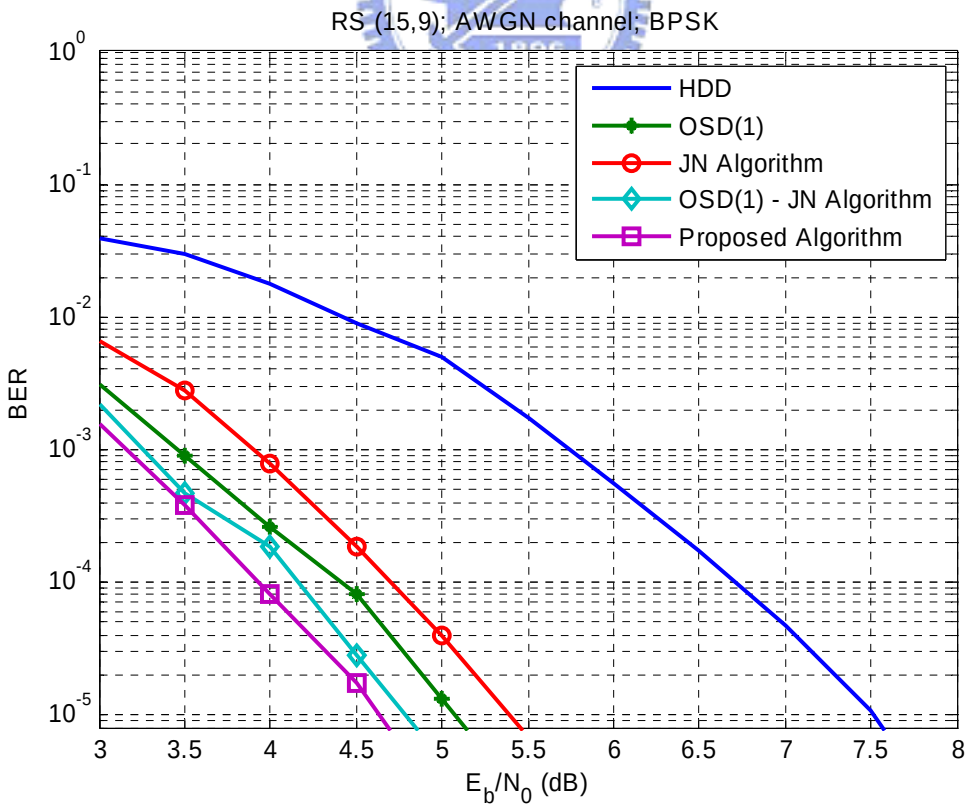


圖 5-7 (15,9) 里德所羅門碼在不同演算法間表現

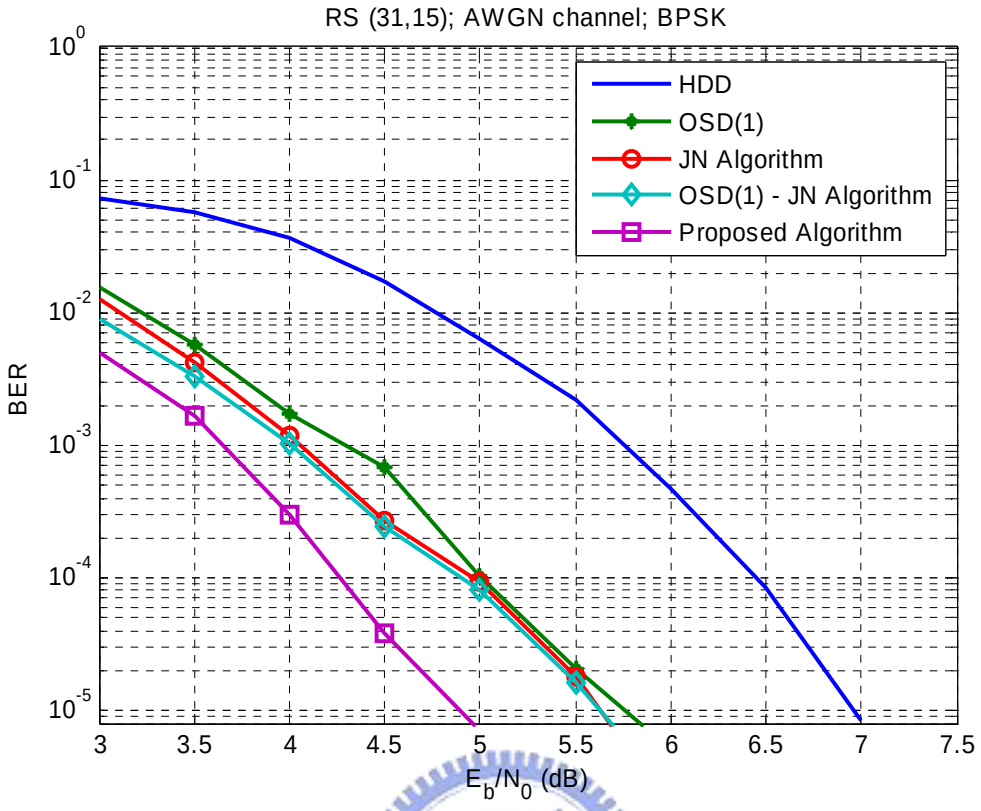


圖 5-8 (31,15) 里德所羅門碼在不同演算法間表現

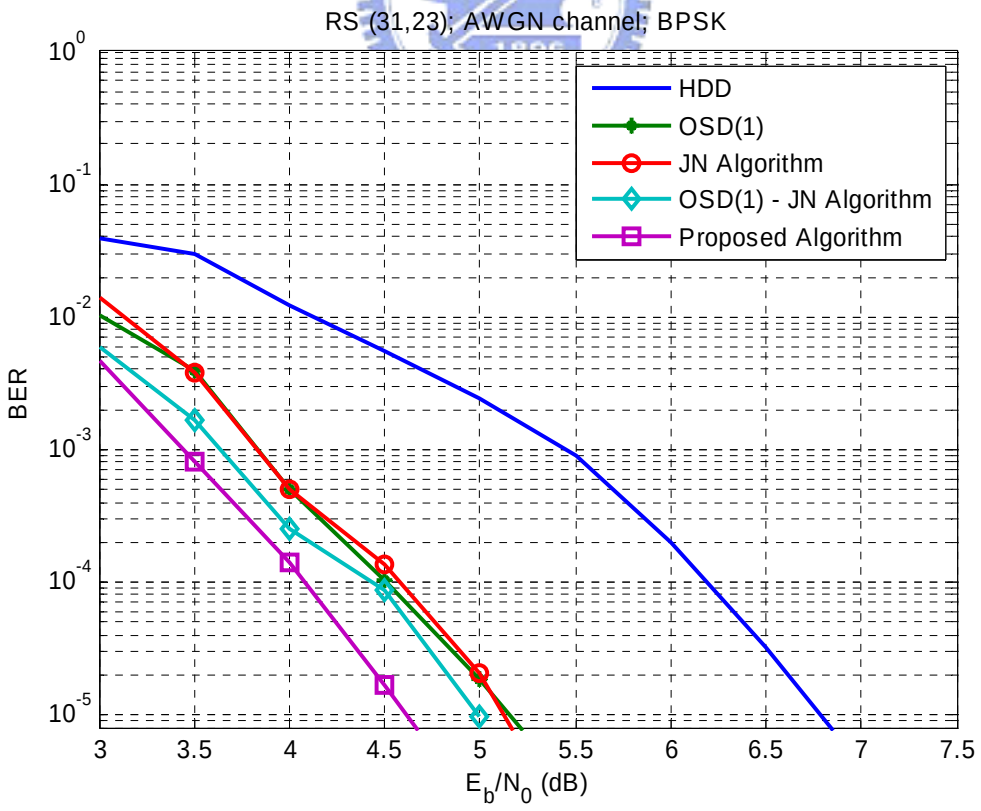


圖 5-9 (31,23) 里德所羅門碼在不同演算法間表現

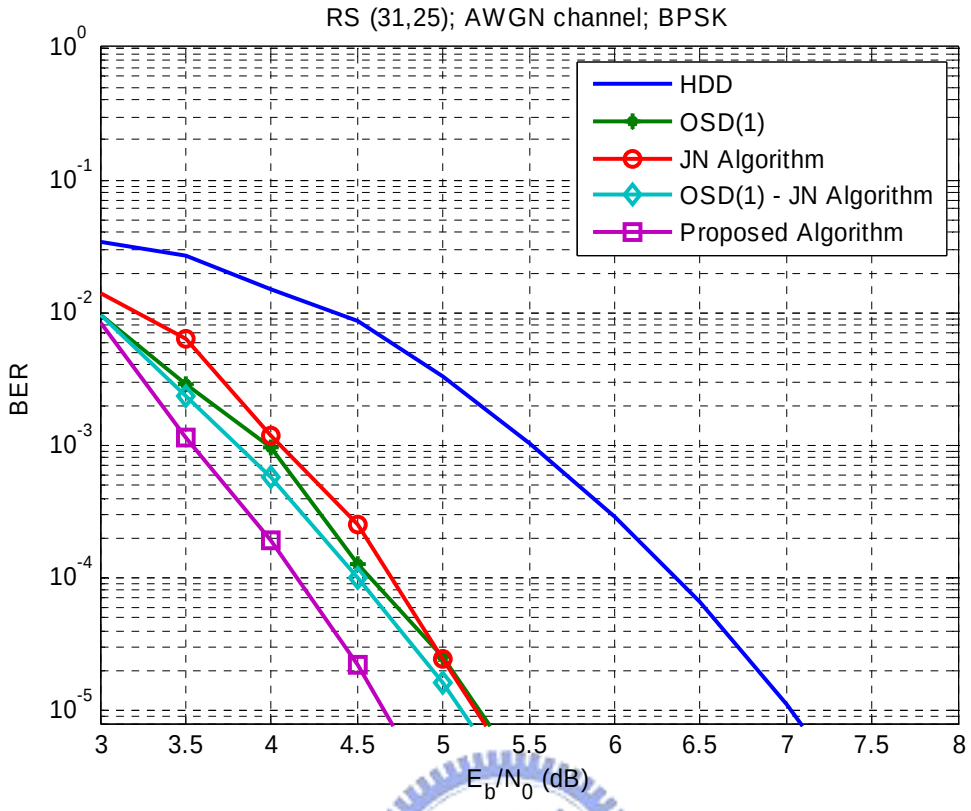


圖 5-10 (31,25) 里德所羅門碼在不同演算法間表現

但由模擬結果可以發現，在效能上提出之演算法較 JN-OSD(1)演算法的確由於解決了高信任度錯誤位置而提供了良好的效能改善。對於提出之演算法較 JN-OSD(1)演算法於每次疊代中增加的運算複雜度，為 $n\log_2 n$ 的排序運算和 $n-k$ 的 LLR 實數加法，其於 (15,7) 里德所羅門碼時，在每次疊代增加了 354 的排序比較運算量和 32 的加法運算量，而於位元錯誤率 10^{-5} 時，得到 0.45dB 的效能改善；而於 (15,9) 里德所羅門碼時，每次疊代增加了 354 的排序比較運算量和 24 的加法運算量，而於位元錯誤率 10^{-5} 時，得到 0.2dB 的效能改善；於 (31,15) 里德所羅門碼時，每次疊代增加了 1128 的排序比較運算量和 80 的加法運算量，而於位元錯誤率 10^{-5} 時，得到 0.6dB 的效能改善； (31,23) 里德所羅門碼時，每次疊代增加了 1128 的排序比較運算量和 40 的加法運算量，而於位元錯誤率 10^{-5} 時，得到 0.4dB 的效能改善； (31,25) 里德所羅門碼時，在每次疊代增加了 1128 的排序比較運算量和 30 的加法的運算量，而於位元錯誤率 10^{-5} 時，得到

0.5dB 的效能改善。一般在文獻上的比較，多採用 (15,7)、(15,9)、(31,15)、(31,25)、(63,55)、(128,64)、(255,239) 等里德所羅門碼。

第六章

結論

在本論文中，一開始介紹了里德所羅門碼的二位元映射方式，利用此映射方法能夠使里德所羅門碼的軟式解碼問題看待成一般二位元線性區塊碼的軟式解碼問題。接下來說明了梯度下降法的理論基礎、運作方式以及其與傳統和積演算法之不同點。於第四章中，我們對於里德所羅門碼所使用的 JN 演算法進行錯誤傳遞問題的討論，從討論中了解高信任度位置錯誤對於 JN 演算法所造成的嚴重影響。

而在本論文中於第四章中提出了適當的回授機制來協助 JN-OSD(1)演算法中，受到高信任度錯誤之影響而造成的問題。此回授機制能夠幫助低信任度位置的錯誤加速被更正，且加速低信任度正確位置的信任度提昇，進而和高信任度位置進行交換的動作。若本來位於二位元奇偶校驗矩陣高密度位置的高信任度位置發生錯誤，則有機會經由此回授機制於疊代過程中，轉而落於奇偶校驗矩陣低密度位置，減少錯誤傳遞的情況。

我們於第四章中亦提出數據上的分析，顯示 JN-OSD(1)演算法受高信任度錯誤的影響，的確在回授機制的運作下獲得了妥善的控制，並且最後以模擬結果說明其效能改善情況。和 JN-OSD(1)演算法相較之下，此回授機制所付出的額外複雜度極低，僅於每次疊代中增加一次 LLR 排序和 LLR 更新的動作，然而得到的效能改善卻相當顯著。此論文的主要貢獻為分析 JN 演算法的潛在問題，並且提出回授機制的改善方式。此概念可應用於相近的演算法中，以解決高信任度錯誤傳遞的影響。

在此方向，可以延續的想法為減少提出之演算法本身的複雜度，並且利用里

德所羅門碼的特性來抑制一些提出之演算法所無法更正的錯誤情況。而是否有其他動作能夠取代較為複雜高斯消去法，而同樣避免掉低信任度位置錯誤的影響，也是可以思考的課題之一。

參考文獻

[1] I. S. Reed, “A class of multiple-error-correcting codes and the decoding scheme,” *IRE Trans.*, vol. IT-4, pp. 38-39, Sep. 1954.

[2] G. D. Forney, Jr., “Generalized minimum distance decoding,” *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 125-131, Apr. 1966.

[3] D. Chase, “A class of algorithms for decoding block codes with channel measurement information,” *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 170-182, Jan. 1972.

[4] M. P. C. Fossorier and S. Lin, “Soft-decision decoding of linear block codes based on ordered statistics,” *IEEE Trans. Inform. Theory*, vol. 41, pp. 1379-1396, Sep. 1995.

[5] A. Vardy and Y. Be’ery, “Bit-level soft-decision decoding of Reed-Solomon codes,” *IEEE Trans. Commun.*, vol. 39, pp. 440-445, Mar. 1991.

[6] V. Ponnampalam and A. Grant, “An efficient SISO algorithm for Reed-Solomon codes,” in *Proc. Int. Symp. Imform. Theory*, Yokohama, Japan, Jul. 2003, pp. 204.

[7] C. Y. Liu and S. Lin, “Turbo encoding and decoding of Reed-Solomon codes through binary decomposition and self-concatenation,” *IEEE Trans. Commun.*, vol. 52, no. 9, pp. 1484-1493, Sep. 2004.

[8] J. Jiang and K. R. Narayanan, “Iterative soft decision decoding of Reed-Solomon codes,” *IEEE Lett. Commun.*, vol. 8, no. 4, pp. 244-246, Apr. 2004.

[9] T. R. Halford and K. M. Chugg, “Random redundant soft-in soft-out decoding of linear block codes,” *in Proc. Int. Symp. Inform. Theory*, Seattle, Washington, Jul. 2006, pp. 2230-2234.

[10] T. Hehn, J. B. Huber, S. Laendner, and O. Milenkovic, "Multiple-bases belief-propagation for decoding of short block codes," *in Proc. Int. Symp. Inform. Theory*, Nice, France, Jun. 2007, pp. 311-315.

[11] J. Jiang and K. R. Narayanan, "Iterative soft decision decoding of Reed-Solomon codes based on adaptive parity-check matrices," *in Proc. Int. Symp. Inform. Theory*, Chicago, Illinois, Jul. 2004, pp. 261.

[12] A. Kothiyal, O.Y. Takeshita, W. Jin and M. Fossorier, "Iterative reliability-based decoding of linear block codes with adaptive belief propagation," *IEEE Lett. Commun.*, vol. 9, no. 12, pp. 1067-1069, Dec. 2005.

[13] A. Kothiyal, O.Y. Takeshita, "A comparision of adaptive belief propagation and the best graph algorithm for the decoding of linear block codes," *in Proc. Int. Symp. Inform. Theory*, Adelaide, Australia, Sep. 2005, pp. 724-728.

[14] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error Correcting Codes*, NY: North Holland, 1977.

[15] J. Jiang and K. R. Narayanan, "Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix", *IEEE Trans. Inform. Theory*, vol. 52, no. 8, pp. 3746-3756, Aug. 2006.

[16] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications, 2nd ed.*. Englewood Cliffs, NJ: Prentice Hall, 2004.

[17] H. Robert and M. Zaragoza, *The Art of Error Correcting Codeing, 2nd ed.*. Canada: John Wiley & Sons, 2002.

[18] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice Hall, 1995.