國立交通大學

電機學院IC設計產業研發碩士班

碩士論文

適用於最小和重組LDPC解碼演算法之補償技術

Compensation Technique for Min-Sum Shuffled LDPC decoding algorithm

研 究 生：陳美宇

指導教授：劉志尉　教授

中 華 民 國九十七年 九 月

適用於最小和重組 LDPC 解碼演算法之補償技術

Compensation Technique for Min-Sum Shuffled LDPC decoding algorithm

研 究 生：陳美宇　　Student：Mei-Yu Chen

指導教授：劉志尉　　Advisor：Chih-Wei Liu

國 立 交 通 大 學

電機學院 IC 設計產業研發碩士班

碩 士 論 文

A Thesis

Submitted to College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Industrial Technology R & D Master Program on

IC Design

September 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年九月

適用於最小和重組 LDPC 解碼演算法之補償技術

研究生：陳美宇　　指導教授：**劉志尉 教授**

國立交通大學電機學院產業研發碩士班

摘　　要

Shuffled BP(belief propagation) algorithm 是一種低密度同位檢查碼(low density parity check，LDPC)的解碼演算法，它的解碼錯誤更正效能高而且解碼遞迴次數收斂快。由於 shuffled BP algorithm 使用了非線性的計算，使得硬體的設計變得十分複雜。針對這個問題，設計者常使用最小項來近似此種複雜的非線性運算，以簡化硬體，此演算法稱為 min-sum shuffled BP algorithm。然而，min-sum shuffled BP algorithm 雖然達到硬體簡化的目的，卻造成解碼錯誤更正效能的下降。為了解決這個問題，本論文探討對 min-sum shuffled BP algorithm 的補償技術，包括了一維，二維 normalization、或 offset 之靜態補償方法，以及動態補償技術等，希望藉由補償技術將 min-sum shuffled BP algorithm 的編碼增益修正，使其達到與傳統的 shuffled BP algorithm 一樣好的效能。我們以 IEEE 802.11n 系統做模擬實驗，模擬結果顯示，經過補償後的 compensated min-sum shuffled BP algorithm，不但保有硬體簡化的特性，其解碼錯誤更正效能也十分接近傳統的 shuffled BP algorithm。

Compensation Technique for Min-Sum Shuffled LDPC decoding algorithm

Student : Mei-Yu Chen Advisor：Dr. Chih-Wei Liu

Industrial Technology R & D Master Program of
Electrical and Computer Engineering College
National Chiao Tung University

ABSTRACT

Shuffled belief propagation (BP) algorithm for the decoding of low-density parity-check (LDPC) codes achieves a remarkable error performance and fast convergence. Nevertheless, it seems to be too complex for hardware implementation. The shuffled BP algorithm can be simplified by using the min-sum approximation, namely the min-sum shuffled BP algorithm; however, the min-sum shuffled BP algorithm suffers from remarkable performance degradation. In this thesis, to solve this problem, we explore some compensation techniques for the min-sum shuffled BP algorithm, including 1D-, 2D-normalization/-offset static schemes and the dynamic scaling approach. Simulations show that the compensated min-sum shuffled BP algorithm achieves the performance very close to that of the original shuffled BP algorithm in IEEE 802.11n system.

誌謝

本篇論文得以完成，首先要感謝我的指導教授－劉志尉老師。在我研究所求學期間，老師在論文研究上給予我詳盡的指導，使我的研究方向不會偏離正軌，老師細心、嚴謹的求學態度更使我獲益匪淺。在這裡，真的是非常感謝老師。

其次，要感謝彥欽學姐、士賢以及一起修課的同學們，在我的研究及課業學習上提供指導與協助。還有感謝口試委員老師－許騰尹老師、桑梓賢老師的指導，使得論文能更豐富與完善。

最後，要特別感謝我的爸爸、媽媽、哥哥以及我的好朋友奕善。他們給予我精神上的支持與鼓勵，陪伴我渡過所有的艱難。在此，我要將我的論文獻給所有關心與輩助我的人。

陳美宇

謹誌於 新竹

2008 年 九月

目錄

表目錄

圖目錄

第一章 簡介

LDPC(Low Density Parity Check) codes 是一種特殊的線性區塊碼，由於它使用具稀疏性的同位檢查矩陣(parity check matrix)，使得編碼與解碼計算複雜度得以降低。在解碼部份，藉由 bit node 與 check node 兩種節點，返複傳遞 sum-product 計算的通道值對數概似比值(log-likelihood ratio，簡稱 LLR)，進行訊息可靠度交換，使用遞迴(iteration)處理方式來逼近謝農極限，稱之 BP decoding (Belief Propagation Algorithm)。這種遞迴處理(Iteration Process)，目的是打亂位元次序，使易錯與不易錯的位元互相混雜，所以若接收機接收到連續的錯誤位元時，迴旋碼則不易正確解回，而 LDPC 卻能輕易解回正確的碼字。

BP decoding 利用完全平行處理資料的方式進行遞迴解碼，雖然達到最佳效能，但缺點是需要大量的遞迴次數[19]。同時，在硬體上需要使用大量的記憶體、處理器及複雜的繞線網路，硬體實現仍然很困難[21]。所以本論文採取重組 BP 解碼(shuffled BP decoding)[19]來改良 BP decoding 的缺點，將硬體實作採取對節點分組的方式，進行序列化的資料處理。這種做法不但可以減少遞迴解碼次數，增加解碼的速度，又可以減少記憶體的使用。

但是，shuffled BP decoding 與 BP decoding 相同，在計算 check node update 時，同樣都面臨複雜的非線性運算 $\ln|\tanh(x)|$，造成硬體設計十分複雜。所以，本論文將 Min Sum decoding [9]使用最小項來近非線性函數 $\ln|\tanh(x)|$計算的觀念，應用到 shuffled BP decoding，為了方便起見，稱之 min-dum shuffled decoding(簡稱 MS shuffled decoding)。由於 MS shuffled decoding 只用到比較器與加法器，大為簡化硬

體的複雜度。但缺點是糾正錯誤能力(error performance)降低，損失編碼增益。

關於 MS shuffled decoding 更正錯誤效能下降的問題，我們仿效在 BP 中，min-sum decoding 使用補償係數來修補下降的解碼效能的作法。將 BP 的補償模式[9] [14][15][16][17]，應用在 MS shuffled decoding 中，為 compensated min-sum shuffled decoding(簡稱 compensated MS shuffled decoding)。

本論文的研究，以 802.11n 做模擬系統的架構，首先使用 shuffled BP decoding 來減少遞迴解碼的次數，改善 BP decoding 須要較多的遞迴次數的缺點；再使用 MS shuffled decoding 來降低 shuffled BP decoding 的硬體複雜度，之後，透過模擬，找到補償係數，藉由補償系數來修正 MS shuffled decoding 造成解碼糾錯效能的損失，達到硬體設計簡單，解碼糾正錯誤效能又能近似 shuffled BP 的目的。

論文的編排如下，第二章對 LDPC 碼做概括性的介紹，從模擬系統的方塊圖功能，與編碼結構到解碼演算法，都有詳盡的描述。第三章介紹 MS shuffled decoding 之補償技術，3.1 節介紹 MS shuffled decoding，它使用 MS shuffled decoding 來簡化 Shuffled BP decoding 的硬體，3.2 節則使用 Compensated MS Shuffled decoding 來提升 error performance，第四章為模擬結果，我們將未補償的 MS Shuffled decoding 與補償後的 Compensated MS Shuffled decoding 以及 Shuffled BP decoding 三種解碼演算法做模擬，以觀察 compensated MS shuffled decoding 的補償效果。大部份的模擬結果顯示 compensated MS shuffled decoding 的 error performance 非常接近 shuffled BP decoding，顯示補償效果十分良好。3.4 節為模擬數據。第五章則是本論文的 summary。

第二章 LDPC codes 介紹

謝農(Shannon)在 1948 年提出謝農極限（Shannon Limit 或 Shannon Bound）[2]，主張以編碼率來定義傳輸通道的傳輸量，他認為任何小於通道容量的編碼率皆可經由這個通道來傳輸。謝農所提出的通道容量限制(Channel Capacity Limit)，可以精準地預測通訊系統的基本極限，而開啟數位通訊大門[22]。

然而麻省理工學院的蓋拉格教授(Gallager)於 1962 年提出一種錯誤修正碼，稱為低密度同位檢查碼(low density parity check code，LDPC code)[2]，因為使用遞迴解碼(iteration decoding)的技術，有很好的編碼增益，相當接近謝農限界，為錯誤更正碼領域帶來一場劃時代的革命。但由於當時超大型積體(VLSI)電路技術尚未成熟，硬體實踐困難，再加上當時錯誤修正碼的主流用途幾乎都是聲音的傳送，編碼長度比較短，發掘不出 LDPC 碼的優點。因為這些原因，LDPC 被束之高閣，也就逐漸被世人遺忘。

近年來由於積體電路設計的進步，使得 LDPC codes 高速平行解碼製作技術得以實現。由於 LDPC codes 的同位檢查矩陣有很少的 1，所以計算量很少，解碼變得簡單又快速(在 LDPC Codes 中主要是記憶體處理)，編碼增益大，而且消耗功率低，再加上 LDPC Codes 專利已經過期，所以很多公司都可以使用而不必付費[22]。縱合以上的優點，使得 LDPC codes 被廣泛地應用在衛星通訊、光纖通訊、個人通訊系統、寬頻行動通訊、ADSL、磁記錄設備、WLAN、即時語音通訊或須要即時數據處理等應用上。而無線網路標準 802.11n、802.16e 也將 LDPC code 列入 IEEE standard 中。

LDPC codes 的同位檢查矩陣，是一個稀疏的二元矩陣，而一列之中非零元素的個數就稱為這一列的列權重，一行之中非零元素的個數稱為這一行的行權重；若每一列的列權重與行權重分別都一樣，則此種 LDPC 是個正規的 LDPC。反之，則稱為非正規。LDPC 的編碼增益是由同位檢查矩陣決定，通常非正規的 LDPC 編碼增益會比正規的還好，但是愈不正規的 LDPC 實作成硬體複雜度會愈高，因此在編碼增益與硬體實作複雜度間取得平衡是一個值得研究的問題。

2.1 LDPC 模擬系統

2.1.1 模擬系統方塊圖

本論文以 IEEE Standard 802.11n [附件 1]之 LDPC codes 做為模擬系統的架構，如圖 1 所示。

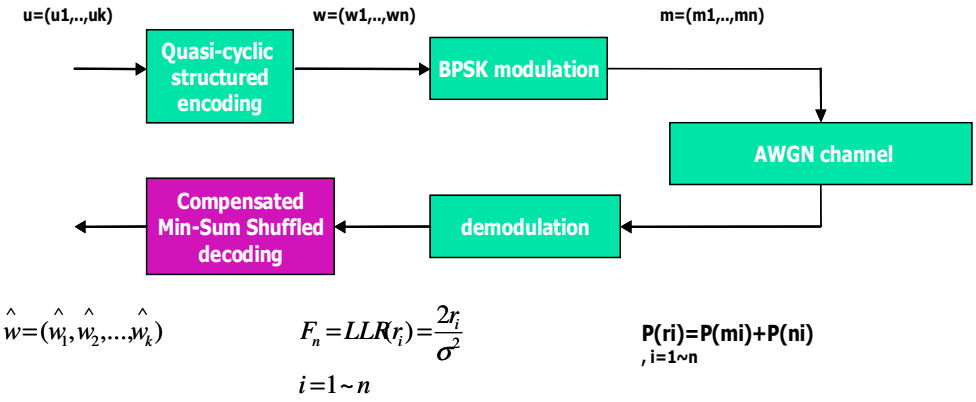


圖 1　LDPC 通訊系統模擬架構圖

訊息位元 u，經過 LDPC 編碼，成為碼字 w。再經由 BPSK 調變以增加抵抗通道雜訊能力[1]；接著送入以高斯雜訊模擬的通道(AWGN channel)。接收端收到混有雜訊與訊號的資料 r 的機率(這裡用機率來表示，因為雜訊是以機率的方式出現。

訊號受雜訊的影響也用機率來表達)。r 的機率經過解調後，以對數概似比值(log-likelihood ratio 簡稱 LLR)來表示，它代表碼字 w 出現 0 的機率與出現 1 的機率的比值再取自然對數；此為 decoding 的輸入，或稱為軟式決策(soft decision)的輸入 soft in；再經過 decoding，得到解碼後的碼字 $\hat{w}$。

2.1.2 系統方塊圖功能簡述

首先介紹編碼，本模擬系統使用準循環結構化編碼(Quasi-cyclic structured encoding)，將長度為 k 的二位元訊息位元 u，編碼成一個長度為 n 的碼字(codeword)w。編碼的方法，請看 2.2 節。

為了增強抗通道雜訊能力，將碼字作 BPSK 調變。BPSK 調變將碼字 w 的 0 調變為 1，1 調變為-1，調變後的碼字以 m 表示，長度為 n[1]。調變完的正負準位，乘上弦波，產生相位相差 180 度的弦形，再送入通道，如圖 2 所示，(a)圖表示調變完的碼字 m，(b)圖表示碼字以相位上相差 180 度的弦波，在通道上傳送。

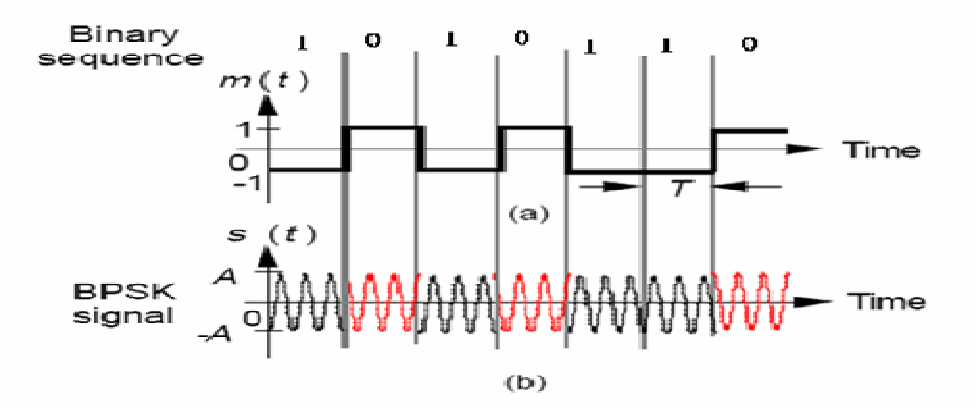


圖 2 BPSK 調變

圖 2(a)與(b)的關係可以用(2-1)式表示:

$$s(t) = A.m(t).\cos 2\pi fct \quad ; \quad 0 \le t \le T \tag{2-1}$$

(2-1)式的 A 表示常數，調變後的 m(t)為 1 或-1，fc 表示載波的頻率，T 為時間間距。

接著，調變後的訊號 m(t)進入雜訊通道。我們使用可加成性的白色高斯雜訊(additive white Gaussian noise，AWGN)，來做為我們的通道雜訊。使用 AWGN 的好處是，因為 AWGN 的可加成性，能獨力影響每個傳輸符號，使得雜訊可以簡單的與訊號做相加，簡化了運算[1]。圖 3 表示高斯雜訊的機率密度函數。σ^2 為高斯雜訊的變異數。

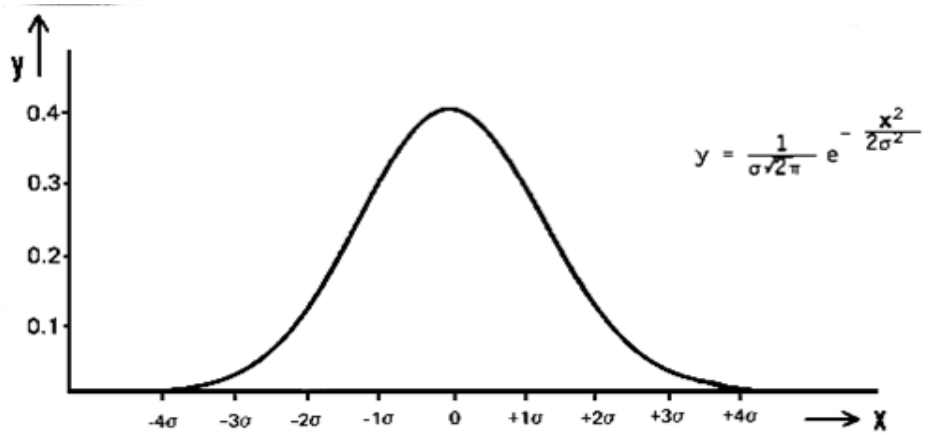


圖 3　高斯雜訊機率密度函數

高斯機率密度函數為 $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{x^2}{2\sigma^2})$，假設 r 是通道所接收的訊息，n 為高斯雜訊；所以訊號 m 經過 AWGN 通道的訊號機率密度為

$$p(r_j) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\frac{-(n_i + m_i)^2}{2\sigma^2} \quad ; \quad m_i = -1 \ \ or \ \ 1 \qquad (2\text{-}2)$$

之後，將訊號做解調。通道所接收的機率訊息 r，經過解調後，得到了 r 的對數概似比值[1]。

公式推導如下:

$$LLR(r_i)=\ln\frac{p(r_i\mid m_i=+1)}{p(r_i\mid m_i=-1)}=\ln\frac{\frac{1}{(2\pi\sigma^2)^{1/2}}\exp[(\frac{r_i+(-1)}{2\sigma^2})^2]}{\frac{1}{(2\pi\sigma^2)^{1/2}}\exp[(\frac{r_i+(-1)^{-1}}{2\sigma^2})^2]}=2\frac{r_i}{\sigma^2}$$

$$\sigma^2=\frac{1}{2R\cdot(\frac{E_b}{N_o})}\quad;\quad R=\frac{u(k\quad bits\quad message)}{n(n\quad bits\quad cordword)}\quad;\quad \frac{E_b}{N_o}=10^{0.1SNR}$$

(2-3)

其中，Eb 為位元能量，No 是雜訊功率頻譜密度。

最後，將訊號進行解碼。解碼器的輸入，為軟式輸入，它接收來自解調器的輸出訊號 LLR 值，也稱為通道值[1]。經過解碼後，得到一個二位元的解碼值 $\hat{w}$，長度為 k。本模擬系統使用 Compensated MS shuffled decoding 做解碼，將於第三章介紹。

2.2 準循環結構化(Quasi-cyclic structured)編碼

由於 LDPC 是一種線性區塊碼，所以可以用訊息向量(message vector)乘上生成矩陣(generator matrix,G)的方式，來產生碼字向量(codeword vector)[1]。但是直接編碼的複雜度很高，與碼長 n^2 成正比。這樣一來，碼字變長時，編碼會變得很複雜。

所以 T.J.Richardson 與 M.A.Shokrollahi 等學者於 2001 年提出了一種有效率的編碼[6]，透過對稀疏性的同位檢查矩陣 H，進行高斯消去法，交換行列的位置，將 H 變成近似下三角形(lower triangular)的型態，對角線以上的部份為 0，以簡化計算，此種過程稱為 preproocess；再將處理過(preprocessing)的 H 分成 6 個次矩陣

ABCDTE，如圖 4 所示，所分成的次矩陣再與碼字(codeword)做運算。這樣把原本複雜的運算拆解成多個較簡單的小區塊的運算，使得每個小區塊運算的複雜度降低成與碼長$n+z^2$，成為線性關係。T.J.Richardson 的編碼，比直接編碼複雜度降低很多。我們可以看出，z 越小，編碼複雜度越低。所以對同位檢驗矩陣進行行列交換時，使 z 盡量小是進一步降低編碼複雜度的關鍵。圖 4 顯示出 H 被劃分的型式[6]。

此外，為了編碼在硬體實作上方便記憶體的定址，再將同位檢查矩陣分成數個以 Zf 為單位的小方陣，每個 Zf 小方陣代表 Zf 單位矩陣經過位移的位移矩陣，此種同位檢查矩陣稱為準循環結構化的同位檢查矩陣(Quasi-cyclic structured parity check matrix)。

如圖 4 所示，H 除了被切割成 6 個矩陣 ABTCDE 之外，還被切割成若干個 Zf 位移矩陣。Zf 位移矩陣上的斜線表示，Zf 單位矩陣經過位移後 1 的位置。在 Standard 802.11n 準循環結構化的同位檢查矩陣元素的數字(請參考附件 1)，代表 Zf 單位矩陣的位移量，而『-』，代表 Zf 位移矩陣為零矩陣。

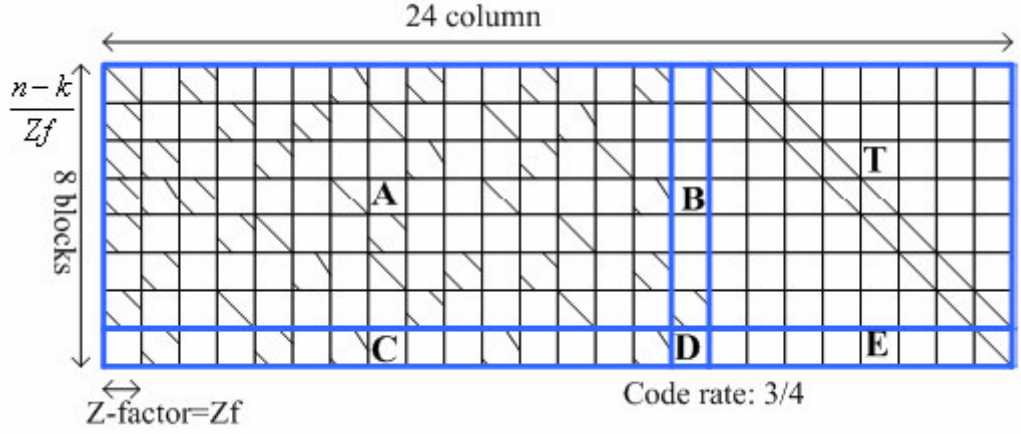


圖 4　準循環結構化編碼的同位檢查矩陣[8]

圖 5 是 Standard 802.16e 的 LDPC 準循環結構化編碼器方塊圖[7]，首先我們先進行編碼公式的推導，再說明編碼步驟。

以下是準循環結構化編碼的公式推導[7]。

u 代表訊息位元，長度為 k；p1 與 p2 代表同位部份(parity part)，p1 長度為 z，p2 長度為 m-z，m 則是冗餘(parity)位元長度，為(n-k)；v 表示編碼完的碼字(codeword)，長度為 n。推導過程如下。

碼字 v 由訊息位元 u，冗餘位元 p1 與 p2 組成。

$$v = [u \quad p_1 \quad p_2] \tag{2-4}$$

同位檢查矩陣切割成 ABCD 次矩陣，再與碼字 v 相乘，以降低計算複雜度。

$$H.v^T = 0$$

$$\begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} . \begin{bmatrix} u \\ p_1 \\ p_2 \end{bmatrix} = 0 \tag{2-5}$$

$$A.u^T + B.p_1^T + T.p_2^T = 0 \tag{2-6}$$

$$C.u^T + D.p_1^T + E.p_2^T = 0 \tag{2-7}$$

由式(2-13)可求得 p_1^T

$$p_2^T = T^{-1}(A.u^T + B.p_1^T) \tag{2-8}$$

將式(2-15)代入(2-14)得到(2-16)

$$C.u^T + D.p_1^T + E.T^{-1}(A.u^T + B.p_1^T) = 0 \tag{2-9}$$

(2-16)經過化簡，得到(2-17)

$$(E.T^{-1}.A + C).u^T + (E.T^{-1}.B + D).p_1^T = 0 \tag{2-10}$$

定義 $\phi = (E.T^{-1}.B + D)$ ， $\phi = I$

由於是二位元運算，所以由(2-17)可以求得 p_1^T

$$p_1^T = (E.T^{-1}.A + C).u^T \tag{2-11}$$

我們將推導出的公式整理如下的編碼步驟。

準循環結構化編碼步驟：

1. 計算 $A.u^T, Cu^T$

2. 計算 $E.T^{-1}(A.u^T)$

3. 計算 p_1^T ，藉由 $p_1^T = (E.T^{-1}.A + C).u^T$

4. 計算 p_2^T ，藉由 $T.p_2^T = A.u^T + B.p_1^T$

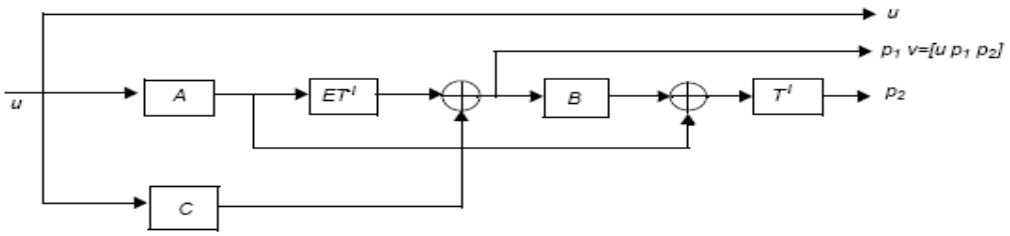


圖 5　802.16e 之 LDPC 準循環結構化編碼器方塊圖[7]

2.3　解碼

1981 年 R.M.Tanner 等學者，提出用 Tanner 圖型的方式來表示 LDPC codes。經過 Mackay 與 Neal 等人重新研究[3]，將 Tanner 圖使用在遞迴解碼演算法(iterative decoding algorithm)上，並證明了 LDPC codes 對於長度較長的碼字，使用 BP 解碼的錯誤更正能力非常接近謝農極限(Shannon limit)[4]。

在介紹 BP decoding 之前，我們先介紹同位檢查矩陣 H。圖 6 代表一個碼長(codeword length)為 6 位元，訊息位元長度為 3，而同位檢查長度為 3 的同位檢查

矩陣(H)。同位檢查矩陣的每一列(row)代表一個 chcek node，簡寫成 c；每一行(column)代表一個 bit node，簡寫成 b。所以，圖 7 的 c1、c2、c3 為 check node1 、check node 2、check node3；而 b1、b2…b6 代表 bit node1、bit node2、…、bit node 6。

$$H = \begin{array}{c} \\ c1 \\ c2 \\ c3 \end{array} \begin{array}{c} \begin{array}{cccccc} b1 & b2 & b3 & b4 & b5 & b6 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array}.$$

圖 6　同位檢查矩陣

bit node 代表解碼器所接收的值，或稱通道值(在這裡，通道值假設為 Fn)。由於 LDPC code 為一線性區塊碼，所以 check node 的值為 bit node 與 H 相乘，再取模組 2 的計算結果，如果 check node 的值為零，代表 bit node 的值為正確的碼字，若不為零，代表錯誤的碼字。

check node 與 bit node 的關係如(2-12)式所描述。

$$H.b^T = 0$$

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = 0$$

$$\begin{aligned} c_1 &= b_1 + b_4 + b_6 = 0 \\ c_2 &= b_1 + b_2 + b_5 = 0 \\ c_3 &= b_2 + b_3 + b_6 = 0 \end{aligned} \qquad (2\text{-}12)$$

為了方便解釋 BP 解碼中 bit node 與 check node 傳遞訊息的情行，我們將同位

檢查矩陣以 Tanner 圖形的方式來表示[5]。同位檢查矩陣 H 對應到 Tanner Graph 的方法為，H 之中元素為 1 的該列 check node 與該行 bit node 會相連接，如圖 7 所示，其中 m 代表 check node 個數，n 代表 bit node 個數。

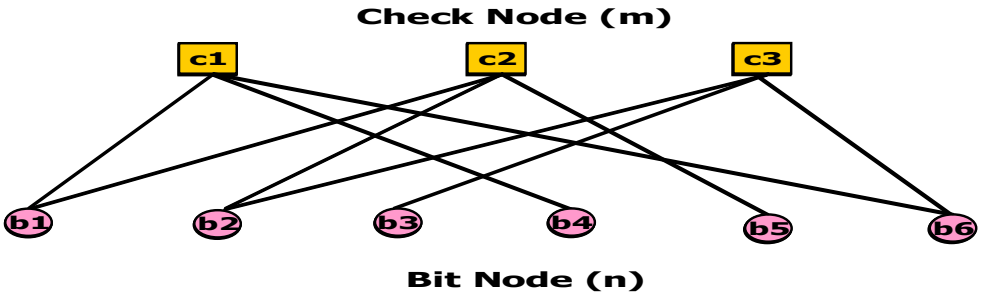


圖 7　Tanner Graph

2.3.1　BP decoding

BP decoding 為一種遞迴解碼[2][4]。BP decoding 依照公式化簡方式的不同，分成蓋拉格法則 BP decoding(Belief Propagation Based on the Gallager's Approach)[9]與 tanh 法則 BP decoding(BP Based on the tanh rule)[9]。蓋拉格法則 BP decoding 可以推導出 min-sum decoding，而 tanh 法則 BP decoding 可以推導出 shuffled BP decoding。

BP decoding，採用軟式遞迴解碼(soft iterative decoding)。藉由 bit node 與 check node 兩種節點，反覆互相傳遞通道訊接收值 Fn 的對數概似比值(log-likelihood ratio，簡稱 LLR)，做 Sum Product 運算來解碼。所以稱為「信度傳播解碼(belief propagation decoding)」，或「訊息傳遞解碼(message passing decoding)」。由於演算法中使用加乘(Sum Product)運算，所以也被稱作加乘演算法解碼(Sum Product algorithm，SPA)[8]。

解碼中所傳遞的 LLR 是一個對數概似比值，它代表訊息為 0 的機率對訊息為 1 的機率的比值，再取自然對數。所以，解碼的結果，如果 bit node 的 LLR 大於 0，就表示訊息為 0 的機率較大，所以被認為是 0；若 LLR 小於 0，則被認為訊息為 1 的機率較大，所以被認為是 1；此過程稱為硬式決策(hard decision)[19]。

2.3.1-1 蓋拉格法則 BP decoding

如同 2.3.1 節所述，蓋拉格法則 BP decoding 與 tanh 法則 BP decoding 都是 BP decoding[9]，這兩種演算法的差別僅在於 check node update 公式的化簡不同而已。

在介紹解碼演算法之前，我們先對系統做一些基本假設與蓋拉格法則 BP decoding 公式所用到的名詞的定義[18]。

假設碼字 $w=(w_1,...w_n)$ 用 BPSK(binary phase shift keying)調變，且傳輸通道雜訊為可加性的白色高斯雜訊(additive white Gaussian noise，AWGN)，雜訊的變異數(variance)為 $N_0/2$。且 $r=(r_1,...,r_n)$ 為通道接收的訊號位元。令 Fn 為位元 n 在解碼器所接收的 LLR 值，$F_n=(4/N_0)r_n$。且令 $E_{mn}^{(i)}$ 是在第 i 次遞迴解碼時，check node 送到 bit node 的位元 n 的 LLR；而 $Z_{mn}^{(i)}$ 是第 i 次遞迴解碼時，bit node 送到 check node 的位元 n 的 LLR。$Z_n^{(i)}$ 代表位元 n 的事後 LLR(posterior LLR)，它為位元 n 解碼後的 LLR。

$H=[H_{mn}]$ 表示，H 是一個 mxn 的二維矩陣，它有 m 個 check node，n 個 bit node。$N(m)=\{n:H_{mn}=1\}$ 代表 Tanner 圖中，與第 m 個 check node 相連接的 bit node 集合；$M(n)=\{m:H_{mn}=1\}$ 則是代表 Tanner 圖中，與第 n 個 bit node 相連接的 check node 集合。

Step 1：位元節點(bit node)與檢查節點(check node)的更新

1.1 位元節點初始化(bit node initialized)

假設 i 為遞迴次數(iteration)，$I_{\max}$ 為最大 iteration。

$$Z_{mn}^{(0)} = F_n \tag{2-13}$$

1.2 檢查節點更新(check node update)

$$E^{(i)}{}_{mn} = Sign \times Magnitude = \{ \prod_{n' \in N(m)\backslash n} \mathrm{sgn}(Z_{mn'}{}^{(i-1)}) \}.\{ \psi^{-1} (\sum_{n' \in N(m)\backslash n} \psi (Z_{mn'}{}^{(i-1)})) \}$$

$n' \in N(m) \backslash n$ 代表與 check node「m」相連接的 bit node 集合，扣除 bit node「n」。

$$\psi(|x|) = \ln |\tanh \frac{|x|}{2}| = \ln \frac{e^{|x|}-1}{e^{|x|}+1} \qquad \psi^{-1}(|x|) = \ln\{\tanh^{-1}(\frac{|x|}{2})\} = \ln \frac{e^{|x|}+1}{e^{|x|}-1}$$

$$if \quad Z_{mn} > 0 \quad , \quad Sign = 1 \quad else \quad Sign = -1 \tag{2-14}$$

1.3 位元節點(bit node update)

$$Z_{mn}^{(i)} = F_n + \sum_{m' \in M(n)\backslash m} E_{m'n}^{(i)} \quad ; \qquad Z_n^{(i)} = F_n + \sum_{m \in M(n)} E_{mn}^{(i)} \tag{2-15}$$

Step 2：硬式決策與停止測試

2.1 硬式決策 (hard decision)

將軟式輸出 LLR「Zn」，經過硬式決策，產生二位元輸出。

$$if \quad z_n > 0 \quad then \quad \hat{w}_n = 0 \quad ; \quad if \quad z_n < 0 \quad then \quad \hat{w}_n = 1$$

2.2 停止遞迴解碼

如果可以滿足 $\hat{w}^{(i)} . H^T = 0$，或已達到所設定的最大遞迴解碼次數 $I_{\max}$，則可以停止遞迴解碼。

Step 3：輸出解出的碼字

$\hat{w}^{(i)}$ 即為我們在第 i 次遞迴(iteration)解出的碼字。

在這裡，我們透過 Tanner 圖來解釋 BP decoding ，請看圖 8~圖 11。

在 Step1.1 中，首先，bit node 用 Fn 通道值對自己做初始化。之後 bit node 傳送通道值 Fn(或解碼器接收的 codeword 的 LLR)給 check node。Step1.2 中，check node 將 bit node 傳來的資料經過 check node update 的計算，扣掉自己的值之後，將 check node update 成 Emn(1)，check node 再傳 Emn(1)給 bit node。Step1.3 中，bit node 將 check node 傳來的資料 Emn(1)與通道值 Fn 做加總，得到新的 bit node update 值 Zn(1)，在此算完成一次遞迴解碼，之後將 bit node Zn(1)的 LLR 值以 hard decision 方式轉換成二位元，如 Step2.1 所示，再與 H 相乘，如果相乘結果為 0 代表 bit node 為正確的 code word，若結果不為 0，則代表解碼錯誤，須再進行下一次遞迴解碼，如 Step2.2 所示。

在下一次遞迴解碼，bit node 扣掉自己的值成為 Zmn(2)，傳給 check node，如 Step1.3 所示。重覆 Step1.2，check node 得到 bit node 送來的 Zmn(2)，經過 check node update 計算再扣掉自己的值，將 check node update 成 Emn(2)，並傳給 bit node。重覆 Step1.3，bit node 將所收到的 check node 值 Emn(2)與 Fn 做加總得到第 2 次遞迴的 bit node update 值 Zn(2)，此算完成第二次遞迴解碼。重覆 Step2.1，將 bit bode Zn(2)轉換成二位元型態並與 H 相乘，若結果不為 0，則須再進行下一次遞迴解碼，如此返覆做遞迴解碼，直到解出正確 codeword 或已達到最大遞迴次數為止，最後再將碼字輸出，如 Step3。

Step 1.1 Initial (Bit node send Fn to check node)

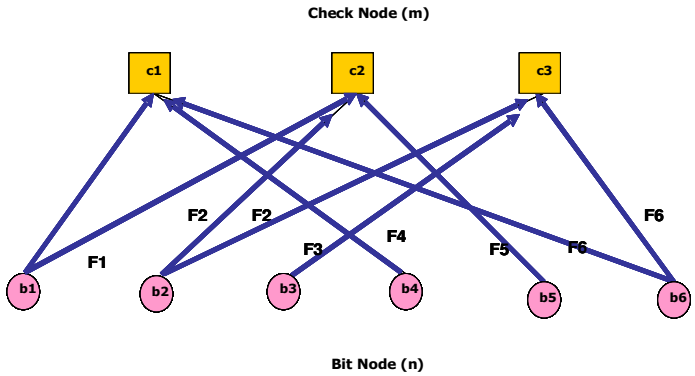


圖 8 BP decoding Step1.1 之 bit node 初始化

Step 1.2 Check node update , then send to bit node

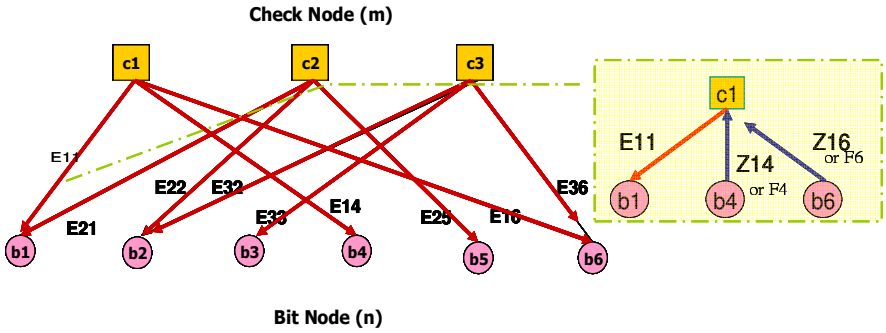


圖 9 BP decoding Step1.2 之 check node update

Step 1.3 Bit node Zn update

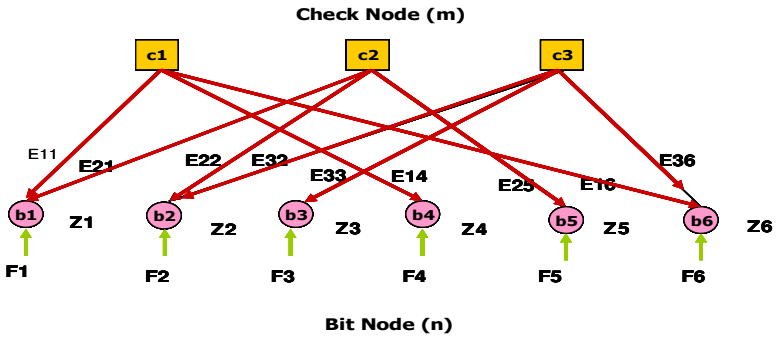


圖 10 BP decoding Step1.3 之 Zn bit node update

Step 1.3 Bit node Zmn update , then send to check node

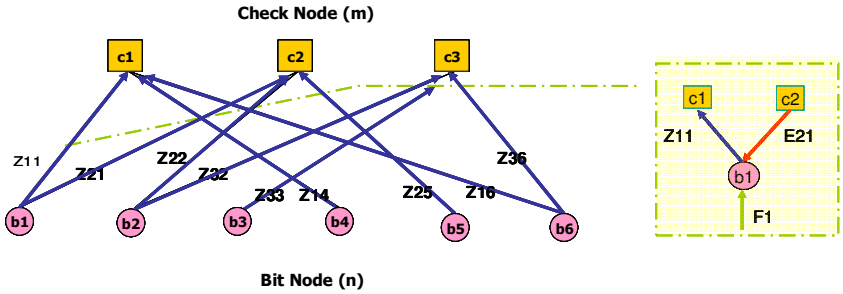


圖 11 BP decoding Step1.3 之 Zmn bit node update

2.3.1-1-1 蓋拉格法則 BP decoding 公式推導

BP decoding 公式推導，為推導 2-16 式中的碼字 w 的 LLR 值。

$$LLR^{posterior}(w_j) = \ln \frac{p(w_j = 0 | r_{j,}, S_j)}{p(w_j = 1 | r_j, S_j)} \quad ; \quad j=1 \sim n \tag{2-16}$$

$LLR^{posterior}(w_j) > 0$，代表 $p(w_j = 0 | r_j, S_j) > p(w_j = 1 | r_j, S_j)$，也就是碼字 $\hat{w}_j$ 為 0 的機率大於 1 的機率，所以判斷 $\hat{w}_j$ =0；反之，若 $LLR^{posterior}(w_j) < 0$，代表 $p(w_j = 0 | r_j, S_j) < p(w_j = 1 | r_j, S_j)$，也就是碼字 $\hat{w}_j$ 為 1 的機率大於 0 的機率，所以認為 $\hat{w}_j$ =1。

假設，碼位元(code bit)是互相獨立的；由貝斯定理(Bayes rule)得到事後機率(a posterior probability，APP)為(2-17)式。

$$p(w_j = b | r_j, S_j) = K \cdot p(r_j | w_j = b) \cdot p(S_j | w_j = b, r_j) \tag{2-17}$$

在式子(2-17)，b 代表 1 或 0；Sj 表示 Tanner 圖中，所有的 check node 連接到第 j 個 bit node 都能滿足同位檢查(parity check)。因為 b 是二位元型態，所以 k 常數可以忽略。

我們將(2-16)式以(2-17)式的 APP 型態展開，得到(2-18)式。

$$LLR(w_j) = \ln \frac{p(w_j = 0 \mid r, s_j)}{p(w_j = 1 \mid r, s_j)}$$
$$= \ln \frac{p(r \mid w_j = 0)}{p(r \mid w_j = 1)} + \ln \frac{p(s_j \mid w_j = 0, r)}{p(s_j \mid w_j = 1, r)} \quad (2\text{-}18)$$

(2-18)式的前項，我們稱為通道值。

通道值的計算如下：

$$\ln \frac{p(r \mid w_j = 0)}{p(r \mid w_j = 1)}$$
$$= \ln \frac{\frac{1}{(2\prod\sigma^2)^{1/2}} \exp \frac{(r_j + (-1)^0)^2}{2\sigma^2}}{\frac{1}{(2\prod\sigma^2)^{1/2}} \exp \frac{(r_j + (-1)^1)^2}{2\sigma^2}} \quad (2\text{-}19)$$
$$= \frac{2r_j}{\sigma^2}$$

(2-18)後項 $\ln \frac{p(s_j \mid w_j = 0, r)}{p(s_j \mid w_j = 1, r)}$ 稱為同位位元項(parity bit term)。

$S_j = \{S_{0j}, S_{1j}, ..., S_{kj}\}$; Smj 代表 Tanner 圖中第 m 個 check node 連接到第 j 個 bit node，滿足同位檢查。同位檢查可以從 2-12 式同位檢查方程式與圖 8 的 Tanner 圖理解，所有 bit node 送到同一個 check node 的 LLR，加總取 2 的餘數(模組 2，module 2)或⊕起來必須為 0。所以 Sj 的集合表示 Tanner 圖中與第 j 個 bit node 相連接的所有 check node 都能滿足同位檢查的集合。所以，同位位元項(parity bit term)可以寫成式 2-20。

$$p(Sj \mid w_j = b, r_j) = p(S_{0j}, S_{1j}, ..., S_{kj} \mid w_j, r_j)$$
$$= \prod_{m \in M(j)} p(S_{mj} \mid w_j = b, r_j) \quad (2\text{-}20)$$

(2-20)式拆開，當 b=0，可以寫成 2-21 式；當 b=1，可以寫成 2-22 式。

parity bit term 公式 ：

$$p(S_j \mid w_j = 0, r_j) = \prod_{m \in M(j)} p(S_{mj} \mid w_j = 0, r_j)$$
$$= \prod_{m \in M(j)} \frac{1}{2}(1 + \prod_{n \in N(m) \setminus j} (q_{mn}^0 - q_{mn}^1)) \quad (2\text{-}21)$$

$$p(S_j \mid w_j = 1, r_j) = \prod_{m \in M(j)} p(S_{mj} \mid w_j = 1, r_j)$$
$$= \prod_{m \in M(j)} \frac{1}{2}(1 - \prod_{n \in N(m) \setminus j} (q_{mn}^0 - q_{mn}^1)) \quad (2\text{-}22)$$

同位位元項 2-21 式與 2-22 式公式推導如下。假設位元 xi 為 1 的機率為 pi，為 0 的機率為 qi。

$p_i = p(x_i = 1)$ ； $q_i = p(x_i = 0) = 1 - p_i$

則兩個位元 x1，x2 做模組 2，會等於 0 或 1。等於 0 時，會產生 2-23 式的結果；等於 1 時，則會等於 2-24 式。

$$p(x_1 \oplus x_2 = 0) = (1 - 2p_1).(1 - 2p_2) \quad (2\text{-}23)$$

$$p(x_1 \oplus x_2 = 1) = -(1 - 2p_1).(1 - 2p_2) \quad (2\text{-}24)$$

假設 L+1 個位元滿足同位檢查的限制。ZL 代表 L+1 個 bit node 連接的 check node，式 2-25 表示，ZL 為 0 時的情形；式 2-26 則表示，ZL 為 1 時的情形。

$$z_L = x_1 \oplus x_2 .... \oplus x_L = 0 \ ; \ z_{L-1} = x_1 \oplus x_2 ... \oplus x_{L-1}$$
$$2p(z_L = 0) - 1 = \prod_{i=1}^{L} (q_i - p_i) \quad (2\text{-}25)$$

$$2p(z_L = 1) - 1 = -\prod_{i=1}^{L} (q_i - p_i) \quad (2\text{-}26)$$

2-27 式表示對單獨一個 check node 連接到多個 bit node 的情行。如圖 12 所示。

$$p(S_{mj}|w_j=0,r_j)=\frac{1}{2}\left(1+\prod_{n\in N(m)\setminus j}(q^0_{mn}-q^1_{mn})\right)$$

$$p(S_{mj}|w_j=1,r_j)=\frac{1}{2}\left(1-\prod_{n\in N(m)\setminus j}(q^0_{mn}-q^1_{mn})\right) \quad (2\text{-}27)$$

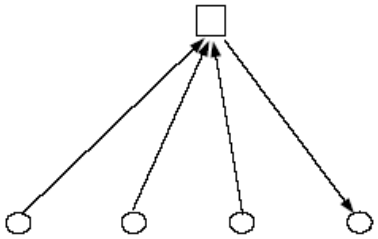


圖 12　單獨一個 check nod 連接到多個 bit node

2-28 式表示對單獨一個 bit node 連接到多個 check node 的情行。如圖 13 所示。

$$\begin{aligned}p(S_j|w_j=0,r_j)&=\prod_{m\in M(j)}p(S_{mj}|w_j=0,r_j)\\&=\prod_{m\in M(j)}\frac{1}{2}(1+\prod_{n'\in N(m)\setminus j}(q^0_{mn'}-q^1_{mn'}))\\p(S_j|w_j=1,r_j)&=\prod_{m\in M(j)}p(S_{mj}|w_j=1,rj)\\&=\prod_{m\in M(j)}\frac{1}{2}(1-\prod_{n'\in N(m)\setminus j}(q^0_{mn'}-q^1_{mn'}))\end{aligned} \quad (2\text{-}28)$$

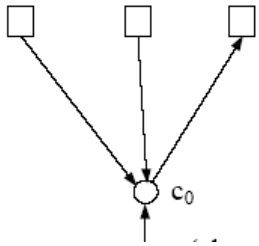


圖 13　單獨一個 bit node 連接到多個 check node

我們將 2-28 式代入 2-18 式，得到下面的 2-29 式。

$$LLR(w_j^0) = \ln \frac{p(w_j = 0 \mid r, s_j)}{p(w_j = 1 \mid r, s_j)} = \frac{2r_j}{\sigma^2} + \sum_{m \in M(j)} \ln \frac{1 + \prod_{n \in N(m) \setminus j} \delta q_{mn}}{1 - \prod_{n \in N(m) \setminus j} \delta q_{mn}} \tag{2-29}$$

$$\prod_i a_i = (\prod_i \operatorname{sgn}(a_i)) * \exp(\sum_i \ln \mid a_i \mid). \tag{2-30}$$

根據 2-30 式的理論，2-29 式可以化成 2-31 式

$$LLR(w_j^0) = \frac{2r_j}{\sigma^2} + \sum_{m \in M(j)} \ln \mid \frac{1 + \prod_{n \in N(m) \setminus j} \operatorname{sgn}(\delta q_{mn}).\exp(\sum_{n \in N(m) \setminus j} \ln \mid \delta q_{mn} \mid)}{1 \_ \prod_{n \in N(m) \setminus j} \operatorname{sgn}(\delta q_{mn}).\exp(\sum_{n \in N(m) \setminus j} \ln \mid \delta q_{mn} \mid)} \mid . \tag{2-31}$$

假設 $\delta q_{mn} = q_{mn}^0 - q_{mn}^1$ ； $LLR(w_{mn}^0) = \ln \frac{q_{mn}^0}{q_{mn}^1}$ ； $\delta q_{mn} = \tanh(\frac{LLR(w_{mn}^0)}{2})$.

(2-32)

從 2-32 式的假設，2-31 式可以化成 2-33 式

$$LLR(w_j^0) = \frac{2r_j}{\sigma^2} + \sum_{m \in M(j)} \ln \mid \frac{1 + s_{mj}.e^{A_{mj}}}{1 - s_{mj} e^{A_{mj}}} \mid \tag{2-33}$$

$$\prod_{n \in N(m) \setminus j} \operatorname{sgn}(\delta q_{mn}) = \prod_{n \in N(m) \setminus j} \operatorname{sgn}[LLR(w_{mn}^0)]$$

$$s_{mj} = \prod_{n \in N(m) \setminus j} \operatorname{sgn}[LLR(w_{mn}^0)]$$

$$A_{mj} = \sum_{n \in N(m) \setminus j} \ln \mid \tanh \frac{LLR(w_{mn}^0)}{2} \mid = \sum_{n \in N(m) \setminus j} \psi(LLR(w_{mn}^0))$$

$$LLR(w_j^0)$$

$$= ChannelValue + \sum_{m \in M(j)} \ln \mid \frac{s_{mj}.e^{A_{mj}} + 1}{s_{mj}.e^{Amj} - 1} \mid$$

當 Smj=1 與-1 時，同位位元項(parity bit term)可以化簡成 2-34 式

If Smj=1

$$\ln|\frac{e^{A_{mj}}+1}{e^{A_{mj}}-1}|=S_{mj}\ln|\tanh^{-1}(\frac{A_{mj}}{2})|$$

If Smj=-1

$$\ln|\frac{-e^{A_{mj}}+1}{-e^{A_{mj}}-1}|=S_{mj}.\ln|\tanh^{-1}(\frac{A_{mj}}{2})| \quad (2\text{-}34)$$

$$LLR(w_j^0)$$

$$=ChannelValue+\sum_{m\in M(j)}\ln|\frac{s_{mj}.e^{A_{mj}}+1}{s_{mj}.e^{Amj}-1}| \quad (2\text{-}35)$$

$$=\frac{2r_j}{\sigma^2}+\sum_{m\in M(j)}S_{mj}\{\psi^{-1}(\sum_{n\in N(m)\setminus j}\Psi(LLR(w_{mn}^0))\}$$

2-35 式可化簡為 2-36 式

$$LLR(w_j^0)$$

$$=\frac{2r_j}{\sigma^2}+\sum_{m\in M(j)}S_{mj}\{\psi^{-1}(\sum_{n\in N(m)\setminus j}A_{mj})\} \quad (2\text{-}36)$$

$$s_{mj}=\prod_{n\in N(m)\setminus j}\mathrm{sgn}[LLR(w_{mn}^0)]$$

$$A_{mj}=\sum_{n\in N(m)\setminus j}\psi(LLR(w_{mn}^0)) \quad (2\text{-}37)$$

$$\psi(x)=\ln|\tanh\frac{x}{2}|=\ln|\frac{e^x-1}{e^x+1}|$$

$$\psi^{-1}(x)=\ln|\tanh\frac{x}{2}|^{-1}=\ln|\frac{e^x+1}{e^x-1}|$$

假設 $Z_{mn}=LLR(w_{mn}^0)$

2-37 式可以用 2-38 式來表示

$$s_{mj}=\prod_{n'\in N(m)\setminus j}\mathrm{sgn}(Z_{mn'}) \qquad A_{mj}=\sum_{n'\in N(m)\setminus j}\psi|Z_{mn'}| \quad (2\text{-}38)$$

同位位元項(check node update)可以用 2-39 式表示，bit node update 可用 2-40 式表示。

所以，導出 check node update 公式(2-39)與 bit node update 公式(2-40)。

Check node update :

$$E^{(i)}{}_{mn} = Sign \times Magnitude = \{ \prod_{n' \in N(m) \backslash n} \mathrm{sgn}(Z_{mn'}{}^{(i-1)}) \}. \{ \psi^{-1} (\sum_{n' \in N(m) \backslash n} \psi (Z_{mn'}{}^{(i-1)})) \} \tag{2-39}$$

Bit node update :

$$Z_{mn} = F_n + \sum_{m' \in M(n) \backslash m} E_{m'n} \tag{2-40}$$

2.3.1-2 tanh 法則 BP decoding

Check node update 推導如下：

tanh 法則 BP decoding 將式子 2-41 代入式 2-42 中，得到式 2-43。

$$\delta q_{mn} = \tanh(\frac{LLR(w^0_{mn})}{2}). \tag{2-41}$$

$$LLR(w^0_j) = \frac{2r_j}{\sigma^2} + \sum_{m \in M(j)} \ln \frac{1 + \prod_{n \in N(m) \backslash j} \delta q_{mn}}{1 - \prod_{n \in N(m) \backslash j} \delta q_{mn}} \tag{2-42}$$

$$LLR(w^0_j) = \frac{2r_j}{\sigma^2} + \sum_{m \in M(j)} \ln \frac{1 + \prod_{n \in N(m) \backslash j} \tanh(\frac{LLR(w^0_{mn})}{2})}{1 - \prod_{n \in N(m) \backslash j} \tanh(\frac{LLR(w^0_{mn})}{2})} \tag{2-43}$$

由於 $LLR(w^0_j) = Z_n$ 且 $LLR(w^0_{mn}) = Z_{mn}$ ， $F_n = \frac{2r_j}{\sigma^2}$ ，又我們假設 $T_{mn} = \prod_{n' \in N(m) \backslash n} \tanh(\frac{Z_{mn'}}{2})$ 。所以，2-43 式可以化簡成 2-44 式。

$$Z_n = F_n + \sum_{m \in M(j)} \ln \frac{1+T_{mn}}{1-T_{mn}} \qquad (2\text{-}44)$$

$$E_{mn}^{(i)} = \ln \frac{1+T_{mn}^{(i)}}{1-T_{mn}^{(i)}}$$

tanh 法則 BP decoding 的檢查節點更新(check node update)如下：

$$T_{mn}^{(i)} = \prod_{n' \in N(m) \backslash n} \tanh\left(\frac{Z_{mn'}^{(i-1)}}{2}\right)$$

$$E_{mn}^{(i)} = \ln \frac{1+T_{mn}^{(i)}}{1-T_{mn}^{(i)}}$$

或

$$E_{mn}^{(i)} = 2\tanh^{-1}(T_{mn}^{(i)}) \qquad (2\text{-}45)$$

2.3.2 min-sum (MS) decoding

BP decoding 公式中的 check node update 部份(見 2-14 式)，因為用到 ln|tanh(x)| 以及 $|\ln\tanh^{-1}(x)|$ 函式，使得硬體的設計十分困難，為了降低硬體的複雜度，M.P.C.Fossorier，M.Mihaljevic 等學者於 1999 年提出[9]，在 check node update 中，使用 bit node 的最小值($\min_{n'\in N(m)\backslash n}\{|Z_{mn'}|\}$)來近似 check node update 的 magnitude 值 ($\psi^{-1}|(\sum_{n'\in N(m)\backslash n}\psi|Z_{mn'}|)|$)；所以，在硬體方面只使用了比較器與加法器，避開 ln|tanh(x)| 以及 $|\ln\tanh^{-1}(x)|$ 函式的使用，使硬體設計大為簡化。後人稱這種演算法為 min-sum decoding(簡稱 MS decoding)或 uniformly most powerful(簡稱 UMP) BP-based decoding[10]。

為了解釋 min-sum decoding ，我們重複之前提到的 $\psi(|x|)$ 函式。

假設， $y=\psi(|x|)=\ln(\tanh\frac{|x|}{2})$ 且 $y'=\psi^{-1}(|x|)=\ln(\tanh\frac{|x|}{2})^{-1}$ ，min-sum decoding 之所以可以用這樣的近似法，是利用 $\psi(|x|)$ 函式的兩個特性[10][11]。

第一個特性是當|x|值越小，對應的|y|值越大，反之亦然，如圖 2-20 之曲線圖。

第二個特性為 $\psi(|x|)$ 與 $\psi^{-1}(|x|)$ 互為反函式。所以 2-46 式成立[10]。

$$\psi^{-1}(\psi(|x|)=|x| \tag{2-46}$$

由於函式 $\psi(|x|)$ 的第一個特性，可知 $\sum_{n'\in N(m)\backslash n}\psi|Z_{mn'}|$ 值會與最大的 $\psi|Z_{mn'}|$ 的值近似，而最大的 $\psi|Z_{mn'}|$ 又會被最小的 $|Z_{mn'}|$ 所決定[10]，故我們可以推得下式：

$$\sum_{n'\in N(m)\backslash n}\psi|Z_{mn'}|\approx\max_{n'\in N(m)\backslash n}\psi|Z_{mn'}|=\psi\{\min_{n'\in N(m)\backslash n}|Z_{mn'}|\} \tag{2-47}$$

又因為函式 $\psi(|x|)$ 的第二個特性，我們將 2-56 式代入 $\psi|\{\sum_{n'\in N(m)\backslash n}\psi|Z_{mn'}|\}|$ 中，並近似為 2-57 式：

$$\psi^{-1}|\{\sum_{n'\in N(m)\backslash n}\psi|Z_{mn'}|\}|$$
$$\approx\psi^{-1}|\psi\{\min_{n'\in N(m)\backslash n}|Z_{mn'}|\}|=\min_{n'\in N(m)\backslash n}|Z_{mn'}| \quad (2\text{-}57)$$

所以，Min Sum decoding ，可以把蓋拉格法則 BP decoding 的 check node update 2-20 式，化簡成 2-58 式。

$$E_{mn}=\{\prod_{n'\in N(m)\backslash n}\mathrm{sgn}(Z_{mn'})\}.\{\min_{n'\in N(m)\backslash n}|Z_{mn'}|\} \quad (2\text{-}58)$$

Min Sum decoding 的 check node update 如下:

$$E_{mn}=\{\prod_{n'\in N(m)\backslash n}\mathrm{sgn}(Z_{mn'})\}.\{\min_{n'\in N(m)\backslash n}|Z_{mn'}|\}$$

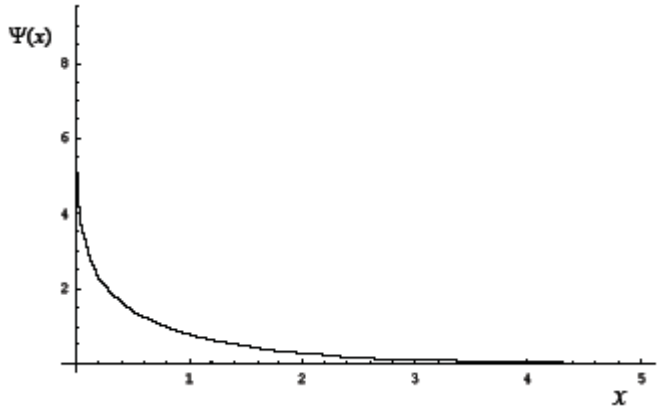


圖 14 $y=\psi(x)$ 函數圖[12]

2.3.3 compensated min-sum (MS) decoding

min-sum decoding 使用最小項近似 BP decoding 的 ln|tanh(x)|，雖然達到降低硬體複雜度，卻造成解碼效能變差。然而這種效能下降，可以使用對 min-sum decoding 的 check node 使用補償係數(正規化 normalization 或偏移量 offset 係數)的方式，來改善[14]。為了方便，本論文稱這種補償方法為 compensated min-sum decoding(簡

稱為 compensated MS decoding)。

對於規則的 LDPC 碼來說，不論是正規化或偏移量補償，只要一個補償係數(1D Compensation factor)來修正 check node，就可以很接近 BP decoding 的效能。但是對於很多不規則的 LDPC 碼來說，只用一個補償係數是不夠的，與 BP decoding 相比，仍然會存在很大的效能損失。所以，J.Zhang， M. Fossorier 等學者於 2005 年提出使用兩個補償係數(2D compensation factor)分別對 check node 與 bit node 做補償，來解決這個問題[15][17]。對於規則的 LDPC 碼，若使用兩個補償係數，會發現它的 check node 與 bit node 的補償係數是相同的[16]。

由於補償係數的不同，所以 compensated min-sum (MS) decoding 又分為 1D-normalized min-sum (MS) decoding、1D-offset min-sum (MS) decoding、2D-normalized min-sum (MS) decoding、2D-offset min-sum (MS) decoding 四種。

2.3.3-1 補償係數推導

我們以 1D-normalized MS decoding 為例，用 density evolution 方式[14]推導 1D-normalized 補償係數。

為了方便推導正規化係數(normalization factor)，我們修改 BPSK 假設與 BP decoding 的 check node update 部份，以及硬式決策方式，如下所示。

假設碼字為 0 時，調變為-1，碼字為 1 時，調變為 1。經過 AWGN 通道，得到 $y_n = s_n + v_n$，v_n 為隨機變數且平均值為 0，變異數為 $N_0/2$。Fn 為解碼器所接收的位元 n 的 LLR 值(或稱通道值)。

BP decoding 的 check node update :

$$T_{mn} = \prod_{n' \in N(m)\backslash n} \frac{1-\exp(Z_{mn'})}{1+\exp(Z_{mn'})} \tag{2-59}$$

$$E_{mn} = \ln \frac{1-T_{mn}}{1+T_{mn}} \tag{2-60}$$

硬式決策：

$$if \quad z_n > 0 \quad then \quad \hat{w}_n = 1 \quad ; \quad if \quad z_n < 0 \quad then \quad \hat{w}_n = 0$$

假設 BP decoding check node update 為 E_{mn}，MS decoding check node update 為 E_{mn}'。2-59 式 BP decoding 的 check node updaet 中，假設 $z = Z_{mn'}$，可以得到 2-61 式

$$\prod_{n' \in N(m)\backslash n} \frac{1-\exp(Z_{mn'})}{1+\exp(Z_{mn'})} = \frac{1-e^{z}}{1+e^{z}} \tag{2-61}$$

令 $|E_{mn}| = |z|$ ； $|E_{mn}'| = \min_{n' \in N(m)\backslash n} |Z_{mn'}|$

$$\frac{1-e^{z}}{1+e^{z}} < \left| \frac{1-\exp(\min_{n' \in N(m)\backslash n} Z_{mn'})}{1+\exp(\min_{n' \in N(m)\backslash n} Z_{mn'})} \right| \tag{2-62}$$

$$\left| \frac{1-e^{x_1}}{1+e^{x_1}} \right| < \left| \frac{1-e^{x_2}}{1+e^{x_2}} \right| \Leftrightarrow |x_1| < |x_2| \tag{2-63}$$

從 2-63 式，可以得知 $|z| < \min_{n' \in N(m)\backslash n} |Z_{mn'}|$，所以我們得到 $|E_{mn}'| > |E_{mn}|$，也就是 MS 的 check node 的 magnitude 值大於 BP 的 check node 的 magnitude，Beta 為正規化係數，代表 BP decoding check node 大小平均值對 MS decoding 的 check node 大小平均值的比值。

$$Beta = \frac{E(|E_{mn}|)}{E(|E_{mn}'|)} \qquad 0 < Beta < 1 \tag{2-64}$$

1D-normalized factor 的理論推導[10][13]：

我們從第一次的遞迴解碼，來求得正規化係數；因為它對整個遞迴解碼的影響最大。

為了便利起見，我們令$\{X_i : i = 1,2,..,W\} = \{Z_{mn'} : n' \in N(m) \setminus n\}$，$W = \lambda - 1$，λ代表每個 check node 與 bit node 相連的個數，Xi 是互相獨立的隨機變數分佈；Xi 的機率密度函數(pdf)與 SNR 值以及編碼(code rate)有關，還有，MS decoding 的初始化，我們假設 Zmn=4/No。

$$E(|E_{mn}|) = E\left(\left|\ln\frac{1-\prod_{i=1}^{W}\frac{1-\exp(X_i)}{1+\exp(X_i)}}{1+\prod_{i=1}^{W}\frac{1-\exp(X_i)}{1+\exp(X_i)}}\right|\right) \tag{2-65}$$

$$E\left(\left|E_{mn}^{'}\right|\right) = E(\min(|X_1|,|X_2|,...,|X_W|)) \tag{2-66}$$

我們求出$E|E_{mn}|$，$E\left|E_{mn}^{'}\right|$；代入 2-64 式便可以求出 Beta。

首先，我們先計算$E\left|E_{mn}^{'}\right|$。令$Y_i = |X_i|, i = 1,2,...,W$；所以，Yi 的 pdf 為 2-67 式所描述。

$$f_{Y_i}(y) = (f_{X_i}(y) + f_{X_i}(-y)u(y) = 2 \cdot f_{X_i}(y)u(y) \tag{2-67}$$

在 2-67 式中，f_{X_i}是 Xi 的 pdf，u(y)是 y 的步階函數。

$$\begin{aligned}\Pr(\left|E_{mn}^{'}\right| > y) &= \Pr(\min(Y_1, Y_2,..., Y_W) > y) \\ &= \Pr\{Y_1 > y, Y_2 > y,..., Y_W > y\} \\ &= [\Pr(Y_1 > y)]^W\end{aligned} \tag{2-68}$$

對於$\left|E_{mn}^{'}\right|>0$，可以得到 2-74 式。

$$E(\left|E_{mn}^{'}\right|)=\int_{0}^{\infty}\Pr(|E_{mn}^{'}|>y)dy \tag{2-69}$$

所以，結合 2-68 與 2-69 式，我們得到

$$\begin{aligned}E(|E_{mn}^{'}|)&=\int_{0}^{\infty}[\Pr(Y_1>y)]^W dy\\&=\int_{0}^{\infty}\left[\int_{y}^{\infty}f_{Y_1}(y_1)dy_1\right]^W dy\\&=\int_{0}^{u}\left[1-Q(\frac{u-y}{\sigma})+Q(\frac{u+y}{\sigma})\right]^W dy\\&\quad+\int_{u}^{\infty}[Q(\frac{y-u}{\sigma})+Q(\frac{y+u}{\sigma})]^W dy\end{aligned} \tag{2-70}$$

其中，$u=\frac{4}{N_0}$，$\sigma^2=\frac{8}{N_0}$ $(\Theta\ LLR^{(0)}(Z_{mn})=\frac{4}{N_0}y_n)$，$Q(x)=\frac{1}{\sqrt{2\pi}}\int_{x}^{\infty}e^{\frac{x^2}{2}}dx$，式 2-70 的第二項積分很小，所以可以被忽略，我們將 2-70 式化簡得到

$$E(\left|E_{mn}^{'}\right|)\approx\int_{0}^{u}[1-Q(\frac{u-y}{\sigma})+Q(\frac{u+y}{\sigma})]^W dy \tag{2-71}$$

為了求得$E(|E_{mn}|)$，我們定義

$$\alpha=\prod_{i=1}^{W}\frac{1-e^{X_i}}{1+e^{X_i}} \tag{2-72}$$

利用泰勒展開式，可推得

$$\ln\frac{1-\alpha}{1+\alpha}=-2(\alpha+\frac{\alpha^3}{3}+\frac{\alpha^5}{5}+...) \tag{2-73}$$

其中，$\alpha,\alpha^3,\alpha^5,...$有相同的正負號(sign)，將 2-65 式以 2-73 式的型式展開，可以得到 2-74 式。

$$E(|E_{mn}|) = E\left(\left|\ln\frac{1-\alpha}{1+\alpha}\right|\right) = 2\sum_{k=1}^{\infty}\frac{E(|\alpha|^{2k-1})}{2k-1} \tag{2-74}$$

令 $m_k = E(|\alpha|^k)$，其中 $\{X_i\}$ 是隨機變數，我們可以推得

$$\begin{aligned} m_k &= E\left(\left|\prod_{i=1}^{W}\frac{1-e^{X_i}}{1+e^{X_i}}\right|^k\right) \\ &= \left[E\left(\left|\frac{1-e^{X_i}}{1+e^{X_i}}\right|^k\right)\right]^W \\ &= \left[E(\tanh(|X_i|/2)^k)\right]^W \\ &= \left[E(\tanh(Y_i/2)^k)\right]^W \end{aligned} \tag{2-75}$$

將 2-75 式代入 2-74 式，可以得到

$$E(|E_{mn}|) = 2(m + m_3/3 + m_5/5 + ...) \tag{2-76}$$

大部份的情形，2-76 式的前幾項就足以代表 $E(|E_{mn}|)$。最後，我們將 2-71 式所求得 $E(|E_{mn}'|)$ 與 2-76 式的 $E(|E_{mn}|)$ 代入 2-64 式，可以得到正規化係數 Beta。

關於遞迴次數跟正規化係數的關係，我們發現用第一次的遞迴計算的正規化係數已經很接近模擬出的正規化係數；我們也試著用不同的遞迴的次數來計算正規化係數，但是發現所得到的正規化係數對解碼的效能改善影響不大。

與推導 1D-normalized 補償係數相同，也可以使用求 density evolution 平均值的方法推導出 1D-offset，2D-normalized，2D-offset 補償係數[14]。

2.3.3-2 1D-normalized MS decoding

由於 1D-normalized MS decoding，是對 MS decoding 的 check node 乘上一個正規化係數 β [14]，用一個補償係數修正 MS decoding，所以 1D-normalized MS decoding 的 check node update 修改如 2-77 式，bit node update 部份與 MS decoding

相同[14][17]。

$$E_{mn}^{(i)} = [\prod_{n' \in N(m)\backslash n} \mathrm{sgn}(Z_{mn'}^{(i-1)})].[\min\{|Z_{mn'}|^{(i-1)}_{n' \in N(m)\backslash n} . \beta\}]$$
$$Z_{mn}^{(i)} = F_n + \sum_{m' \in M(n)\backslash m} |E_{m'n}|^{(i)} \quad ; \quad Z_n^{(i)} = F_n + \sum_{m \in M(n)} E_{mn}^{(i)} \tag{2-77}$$

2.3.3-3 1D-offset MS decoding

由於 1D-offset MS decoding，是對 MS decoding 的 check node 減去一個偏移值 α[13]，用一個補償係數修正 MS decoding，所以 1D-offset MS decoding 的 check node update 修改如 2-78 式，bit node update 部份與 MS decoding 相同[13]。

$$E_{mn}^{(i)} = [\prod_{n' \in N(m)\backslash n} \mathrm{sgn}(Z_{mn'}^{(i-1)})].[\min\{|Z_{mn'}|^{(i-1)}_{n' \in N(m)\backslash n} - \alpha\}]$$
$$Z_{mn}^{(i)} = F_n + \sum_{m' \in M(n)\backslash m} |E_{m'n}|^{(i)} \quad ; \quad Z_n^{(i)} = F_n + \sum_{m \in M(n)} E_{mn}^{(i)} \tag{2-78}$$

2.3.3-4 2D-normalized MS decoding

2D-normalized MS decoding 是將 MS decoding 的 chcek node 乘上一個 β_1 的正規化係數，對 MS decoding 的 bit node 乘上另一個 β_2 的正規化係數，用兩個補償係數分別修正 check node 與 bit node[16][17]，所以 2D-normalized MS decoding 的 check node update 與 bit node update 如 2-79 式所示[16][17]。

$$E_{mn}^{(i)} = [\prod_{n' \in N(m)\backslash n} \mathrm{sgn}(Z_{mn'}^{(i-1)})].[\min\{|Z_{mn'}|^{(i-1)}_{n' \in N(m)\backslash n} . \beta_1\}]$$
$$Z_{mn}^{(i)} = F_n + \sum_{m' \in M(n)\backslash m} \{|E_{m'n}|^{(i)} . \beta_2\} \quad ; \quad Z_n^{(i)} = F_n + \sum_{m \in M(n)} E_{mn}^{(i)} \tag{2-79}$$

2.3.3-5 2D-offset MS decoding

2D-offset MS decoding 是對 MS decoding 的 chcek node 減去一個α_1的偏移係數，對 MS decoding 的 bit node 減去一個α_2的偏移係數，用兩個偏移係數分別修正 check node 與 bit node[16]，所以 2D-offset MS decoding 的 check node update 與 bit node update 如 2-80 式所示[16]。

$$E_{mn}^{(i)} = [\prod_{n' \in N(m)\backslash n} \mathrm{sgn}(Z_{mn'}^{(i-1)})].[\min\{\left|Z_{mn'}\right|_{n' \in N(m)\backslash n}^{(i-1)} - \alpha_1\}]$$
$$Z_{mn}^{(i)} = F_n + \sum_{m' \in M(n)\backslash m}\{|E_{m'n}|^{(i)} - \alpha_2\} \;;\; Z_n^{(i)} = F_n + \sum_{m \in M(n)} E_{mn}^{(i)} \qquad (2\text{-}80)$$

2.3.4 shuffled BP decoding

BP decoding 使用 node 不分組的方式進行解碼，須要較多遞迴解碼次數，運算上使用大量的記憶體，處裡器與複雜的連接繞線技術，使得硬體複雜度提高[21]。為了減少遞迴解碼次數、簡化硬體的設計，J.Zhang 提出了將 node 分組進行解碼[19]，以遞迴次數減少，甚至到減半[20]，這種採用分組的解碼方式，稱為 shuffled BP decoding[18][19]。

分組的方式有兩種[5][21]，一種是將同位檢查矩陣以垂直方法分組(vertical partitioning)，另一種是將同位檢查矩陣以水平方法分組(horizontal partitioning)[21]。兩種方法都能使遞迴次數減少，解碼的錯誤更正效能也相似[5][21]。在此，我們以垂直分組方法來進行研究，也以垂直分組的方法汎指 shuffled BP decoding。

首先，我們先介紹 shuffled BP decoding 的公式，之後再以 H 與 Tanner 圖來解說解碼過程。

假設，碼字長度為 N，將 N 個 bit node 分成 G 組，所以每一組有 N_G 個 bit nodes。$N_G = N/G$ (且 N mod G=0)。

Step 1 : bit node update and check node update

1.1 bit node initialized

假設 i 為遞迴次數(iteration)，$I_{\max}$ 為最大 iteration。bit node 在第 0 次遞迴時，被初始化為(2-81)式。

$$F_n = \frac{2r_j}{\sigma^2} \quad ; \quad Z_{mn}^{(0)} = F_n \tag{2-81}$$

1.2 Horizontal step : check node update

For $1 \le g \le G$ ， $m \in M(n)$

$$T_{mn}^{(i)} = \prod_{\substack{n' \in N(m) \setminus n \\ n' \le (g-1).N_G}} \tanh(\frac{Z_{mn'}^{(i)}}{2}) \prod_{\substack{n' \in N(m) \setminus n \\ n' > (g-1).N_G}} \tanh(\frac{Z_{mn'}^{(i-1)}}{2}) \tag{2-82}$$

$$E_{mn}^{(i)} = \ln \frac{1+T_{mn}^{(i)}}{1-T_{mn}^{(i)}} \tag{2-84}$$

(2-82)式表示 check node update 時，會依照組別的不同，來決定使用同次(i)或上一次(i-1)的 bit node 值。在自己這一組之前的 $n' \le (g-1).N_G$，使用同一次遞迴(i)的 bit node 新資料 $Z_{mn}^{(i)}$，在自己這組之後的 $n' > (g-1).N_G$，使用上一次遞迴(i-1)的 bit node 舊資料 $Z_{mn}^{(i-1)}$，來計算 $T_{mn}^{(i)}$。

1.3 Vertical step : bit node update

$m \in M(n)$

$$Z_{mn}^{(i)} = F_n + \sum_{m' \in M(n) \setminus m} E_{m'n}^{(i)} \qquad Z_n^{(i)} = F_n + \sum_{m \in M(n)} E_{mn}^{(i)} \tag{2-85}$$

Step 2 : 硬式決策與停止測試

2.1 硬式決策 (hard decision)

將軟式輸出 LLR「Zn」，經過硬式決策，產生二位元輸出。

$if \quad z_n > 0 \quad then \quad \hat{w}_n = 0 \quad ; \quad if \quad z_n < 0 \quad then \quad \hat{w}_n = 1$

2.2 停止遞迴解碼

如果可以滿足 $\hat{w}^{(i)} . H^T = 0$，或已達到所設定的最大遞迴解碼次數 $I_{\max}$，則可以停止遞迴解碼。

Step 3 : 輸出解出的碼字

$\hat{w}^{(i)}$ 即為我們在第 i 次遞迴(iteration)解出的碼字。

以上就是 shuffled BP decoding 公式，現在我們以前面的(6,3)code 的同位檢查矩陣為例，以圖 15 與圖 16 來說明 shuffled BP decoding 過程。

shuffled BP decoding 將 H 做垂直分組，以每個 bit node 為一組，共分成 codeword 長度的組數，在此為 6 組，如圖 15 所示。

黃色實線框代表正在進行 node update 的組數，紅色虛線框代表 bit node 在第 i 次遞迴解碼的新值，而藍色虛線代表 bit node 在第 i-1 次遞迴解碼的舊值。

Shuffled BP decoding 以一組進行完 check node update 與 bit node update 之後，再進行下一組的 check node update 與 bit node update 的方式來解碼。

由於上一組的 bit node update 完的新資料，可以做為下一組 check node update 的 bit node 輸入值，所以隨著愈多組數 bit node 與 check node 被更新，愈後面的組數在計算 check node update 時，就可以拿到更多前面組數所產生的 bit node 新值，如圖 15 所示，愈前面的組數計算 check node update，會拿到比較多 i-1 次遞迴解碼的 bit node 舊值(藍色虛線框)來做計算，相反地，愈後面的組數，計算 check node update 時，可以拿到愈多 i 次遞迴解碼的 bit node 新值(紅色虛線框)來做計算，所以隨著愈多組數被 update，資料的更新愈快速。也因此 shuffled BP decoding 可以用較少的遞迴次數，解出正確的碼字。

而 BP decoding 以所有的 bit node 為一組的不分組的方式，進行解碼。必須等到所有的 check node 都 update 完，才能進行 bit node update，所以 check node 只能使用上次遞迴 i-1 解碼的舊 bit node 資料(藍色虛線)，由於資料更新慢，所以須要比 shuffled BP decoding 更多的遞迴次數才能解出正確碼字。

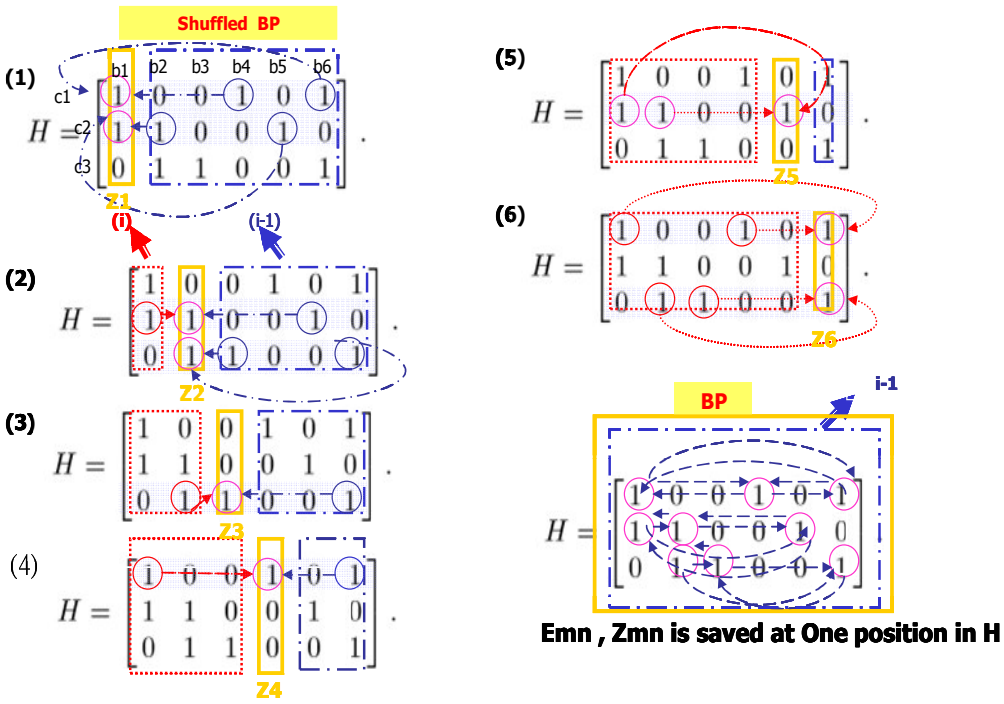


圖 15　shuffled BP decoding 與 BP decoding 的比較

圖 16 用 Tanner graph 來描述 shuffled BP decoding 的解碼過程。與圖 15 使用 H 描述解碼過程相同，也是第一組做完 check node update 與 bit node update，再做下一組的 check node update 與 bit node update，直到 6 組都做完為止，才算完成一次遞迴解碼。藍色虛線同樣代表 i-1 次遞迴 bit node 舊值，紅色虛線代表 i 次遞迴 bit node 新值，而使用色塊的 node 表示正在 update 的 node。

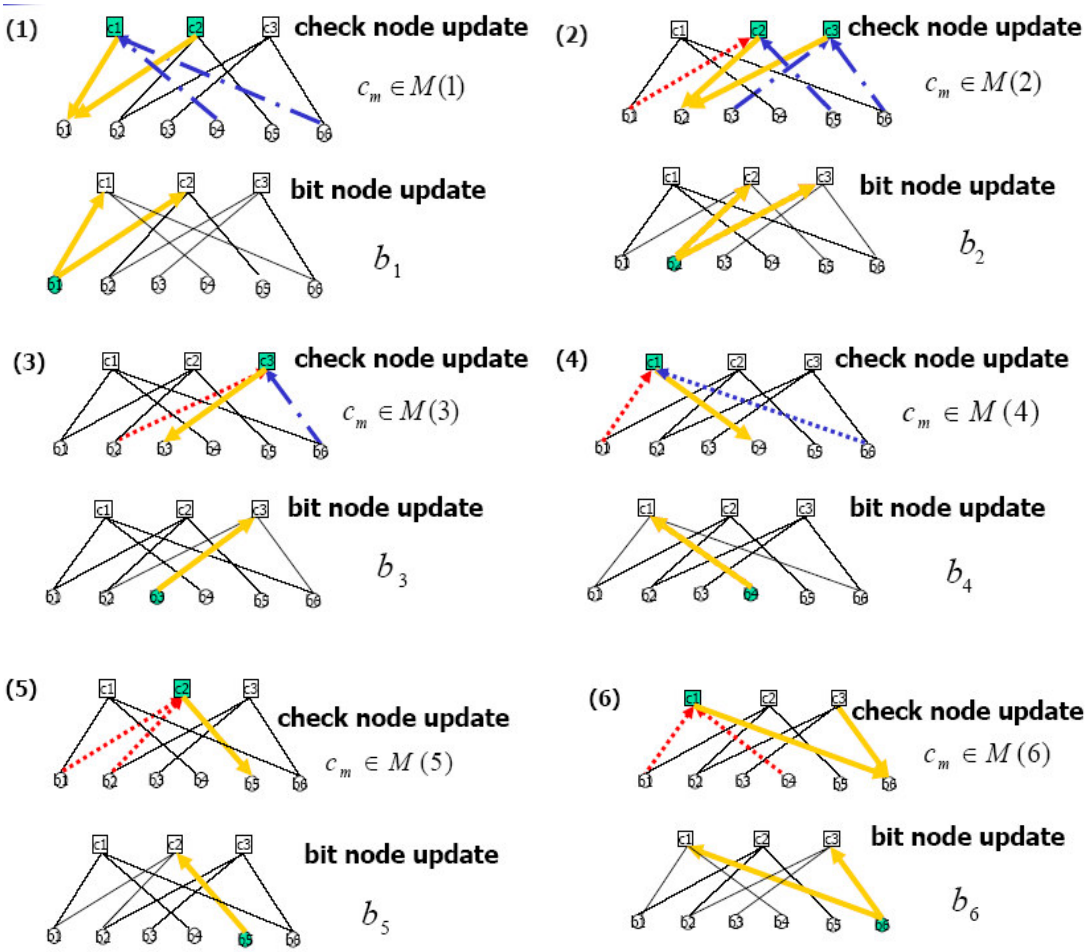


圖 16　Tanner graph 描述 shuffled BP decoding 解碼過程

2.3.4-1　shuffled BP decoding 與 BP decoding 的比較

在這一節，我們主要探討 shuffled BP decoding 與 BP decoding 兩種演算法的差異。首先，我們用解碼流程圖來看兩種演算法的差別，如圖 17 所示；接著，我們將兩種演算法做模擬，比較他們的解碼錯誤更正效能以及遞迴解碼次數，如圖 18、19 所示。

圖 17 為 shuffled BP decoding 與 BP decoding 的解碼過程。它們的解碼過程相同，差別在於 check node update $T_{mn}^{(i)}$ 的不同。BP decoding 的 check node update 全部使用前一次遞迴的 bit node 值 $Z_{mn}^{(i-1)}$ 來計算，Shuffled BP decoding 則會依照組數

的不同，決定使用前一次 bit node 值 $Z_{mn}^{(i-1)}$ 或這一次的 bit node 值 $Z_{mn}^{(i)}$，來計算 check node update。

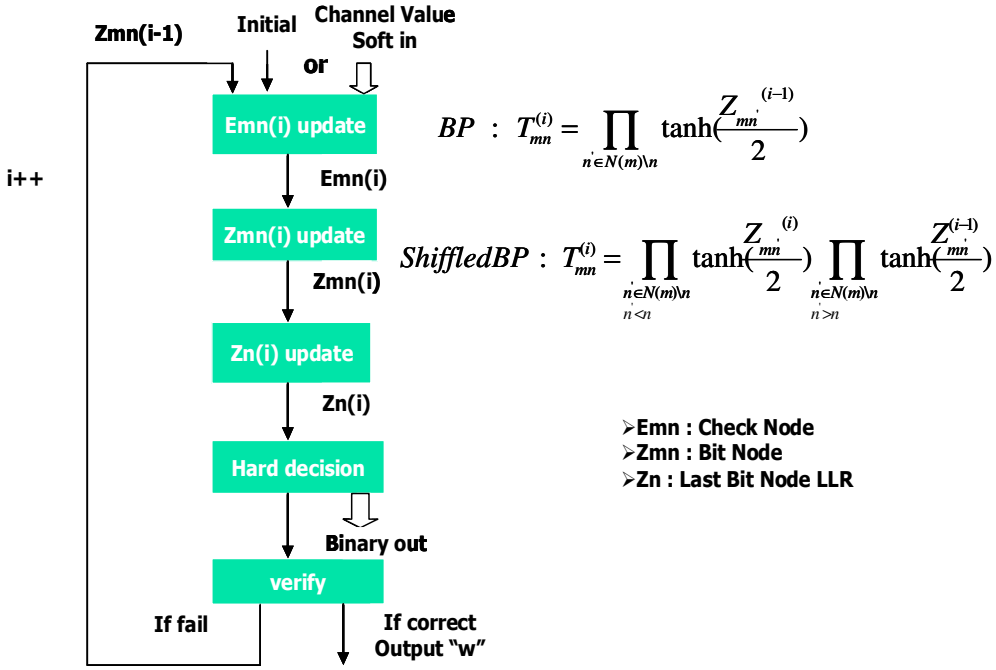


圖 17　shuffled BP decoding 與 BP decding 之差異

圖 18 代表碼長為 648，code rate 為 1/2、2/3、3/4、5/6，最大遞迴次數為 15 次的 shuffled BP decoding 與 BP decoding 的解碼更正錯誤的效能比較圖。

從圖 18 中，我們可以看出，shufled BP decoding 都比 BP decoding 的解碼錯誤率小，表示採用分組的 shuffled BP decding 確實比不分組的 BP decoding，解碼錯誤效能(error performance)來得好。

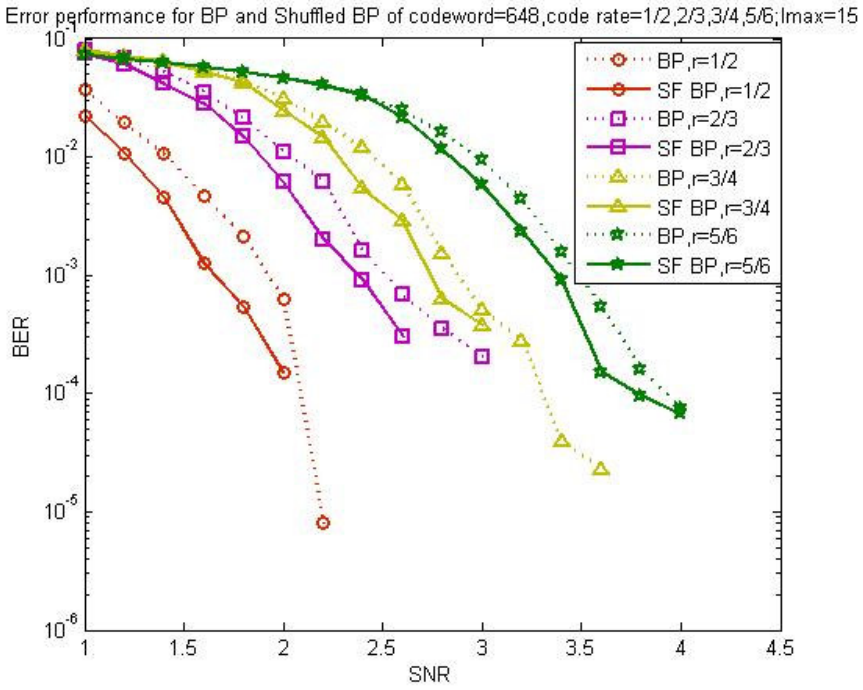
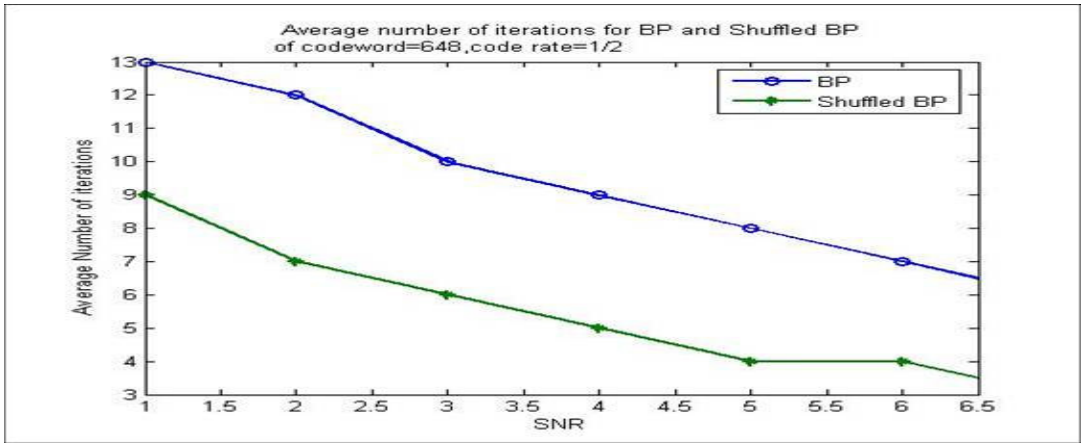


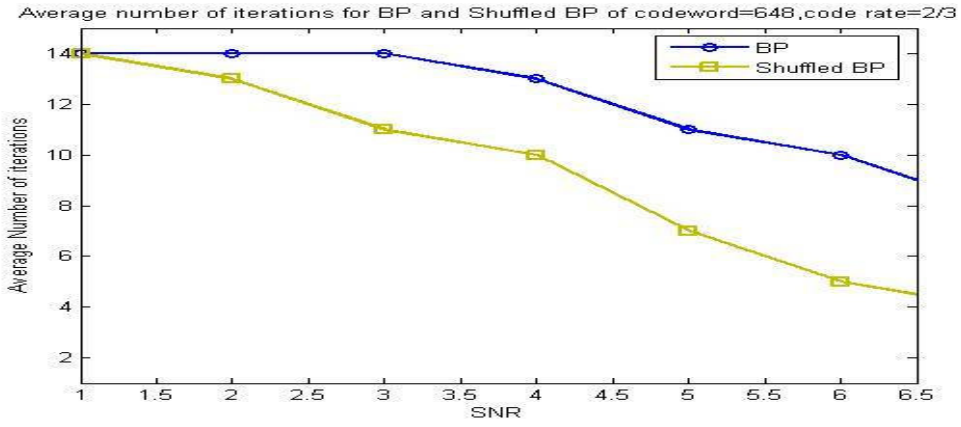
圖 18　shuffled BP decoding 與 BP decoding 之模擬結果

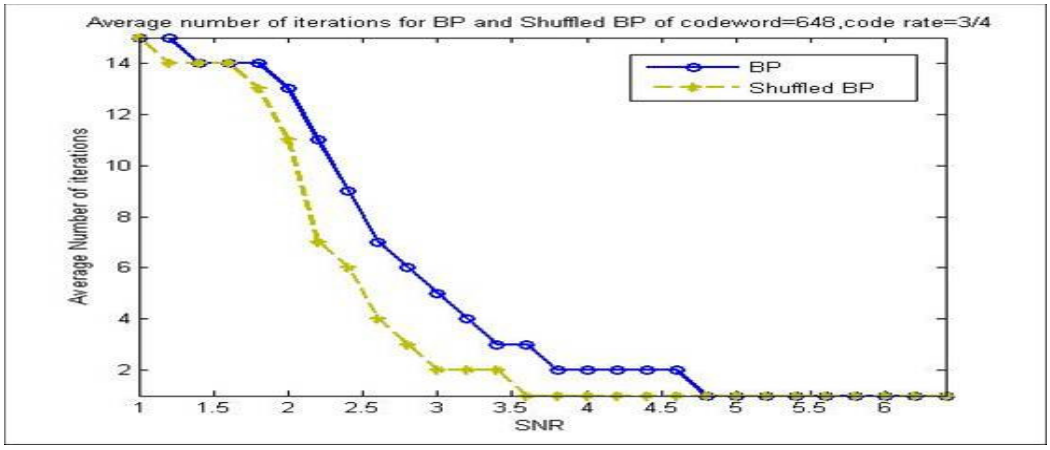
圖 19 代表碼長為 648，code rate 為 1/2、2/3、3/4、5/6 最大解碼遞迴次數(Imax)為 15，shuffled BP decoding 與 BP decoding 在不同 SNR 的解碼遞迴(iteration)次數比較圖。

從圖 19，我們看到，在相同的 SNR 值之下，shuffled BP decoding 的解碼遞迴(iteration)次數都比 BP 的少，與理論推導結果符合。經過統計，我們得到碼長為 648，code rate 為 1/2、2/3、3/4、5/6 的 shuffled BP decoding 平均遞迴解碼次數為 BP decoding 平均遞迴解碼次數的 57%~89%。

此外，透過圖 19，我們也觀察到 shuffled BP decoding 在 code rate 較小時，遞迴解碼次數比 BP decoding 下降較多，隨著 code rate 的增加，兩種解碼的遞迴解碼次數會逐漸拉近。







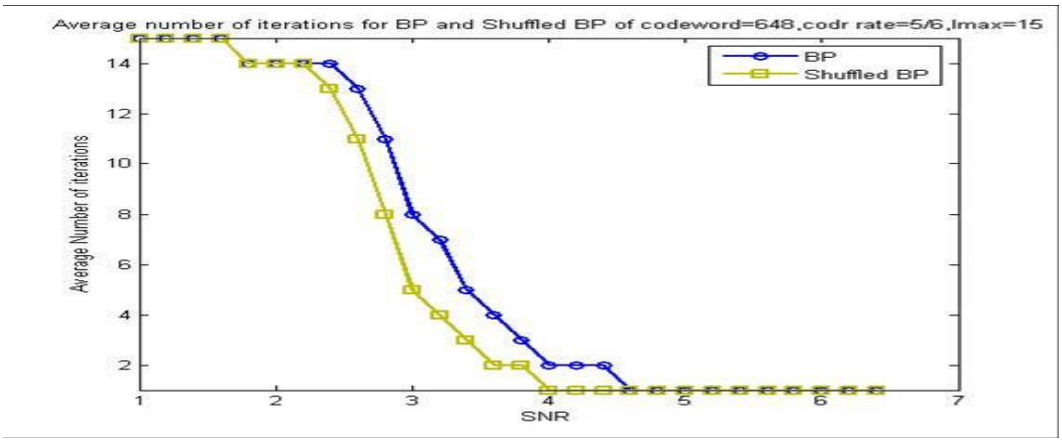


圖 19　shuffled BP decoding 與 BP decoding 解碼遞迴次數之模擬結果

第三章 compensated min-sum (MS) shuffled decoding

3.1 min-um (MS) shuffled decoding

與 MS decoding 相同，計算 check node update 時，MS Shuffled decoding 也是取 bit node 的最小值來近似 shuffled BP decoding 的大小(magnitude)部份，但是與 MS decoding 不同的地方是，MS shuffled decoding 在計算 check node update 時，所用的 bit node 值會有新值舊值($Z_{mn'}^{(i)}, Z_{mn'}^{(i-1)}$)之分，而 MS decoding，計算 check node update 用的都是上一次遞迴的舊 bit node 值 $Z_{mn'}^{(i-1)}$。

式 3-1 為 MS shuffled decoding 的 check node update，除了 check node update 之外，其餘部份都與 shuffled BP decoding 相同。

$$E_{mn}^{(i)} = [\prod_{n' \in N(m)\backslash n} \mathrm{sgn}(Z_{mn'}^{(i)or(i-1)}].[\min_{n' \in N(m)\backslash n} \{| Z_{mn'}^{(i)or(i-1)} |\}] \tag{3-1}$$

3.2 compensated MS shuffled decoding

與 compensated MS decodin 相同，compensated MS shuffled decoding 可以分成 1D-normalized MS shuffled decoding 、 1D-offset MS shuffled decoding 、 2D-normalized MS shuffled decoding、2D-offset MS shuffled decoding。

3.2.1 1D-normalized MS shuffled decoding

1D-normalized MS shuffled decoding 是將 MS shuffled decoding 的 chcek node

乘上一個正規化係數β來做修正，用一個補償係數修正 MS shuffled decoding，1D-normalized MS shuffled decoding 的 check node update 修改成式 3-2，而 bit node update 則維持不變。

$$E_{mn}^{(i)} = [\prod_{n' \in N(m)\backslash n} \mathrm{sgn}(Z_{mn'}^{(i)or(i-1)})].[\min\{|Z_{mn'}|^{(i)or(i-1)}_{n' \in N(m)\backslash n} .\beta\}]$$
$$Z_{mn}^{(i)} = F_n + \sum_{m' \in M(n)\backslash m} \{|E_{m'n}|^{(i)}\} \;;\; Z_n^{(i)} = F_n + \sum_{m \in M(n)} E_{mn}^{(i)} \quad (3\text{-}2)$$

3.2.2 1D-offset MS shuffled decoding

1D-offset MS shuffled decoding，是對 MS shuffled decoding 的 check node 減去一個偏移值α，用一個補償係數修正 MS shuffled decoding，所以它的 check node update 修改如 3-3 式，bit node update 部份維持不變。

$$E_{mn}^{(i)} = [\prod_{n' \in N(m)\backslash n} \mathrm{sgn}(Z_{mn'}^{(i)or(i-1)})].[\min\{|Z_{mn'}|^{(i)or(i-1)}_{n' \in N(m)\backslash n} - \alpha\}]$$
$$Z_{mn}^{(i)} = F_n + \sum_{m' \in M(n)\backslash m} |E_{m'n}|^{(i)} \;;\; Z_n^{(i)} = F_n + \sum_{m \in M(n)} E_{mn}^{(i)} \quad (3\text{-}3)$$

3.2.3 2D-offset MS shuffled decoding

2D-offset MS shuffled decoding，是對 MS shuffled decoding 的 chcek node 減去一個α_1偏移係數，對 MS shuffled decoding 的 bit node 減去一個α_2偏移係數，用兩個偏移係數分別修正 check node 與 bit node，請看 3-4 式。

$$E_{mn}^{(i)} = [\prod_{n' \in N(m)\backslash n} \mathrm{sgn}(Z_{mn'}^{(i)or(i-1)})].[\min\{|Z_{mn'}|^{(i)or(i-1)}_{n' \in N(m)\backslash n} - \alpha_1\}]$$
$$Z_{mn}^{(i)} = F_n + \sum_{m' \in M(n)\backslash m} \{|E_{m'n}|^{(i)} - \alpha_2\} \;;\; Z_n^{(i)} = F_n + \sum_{m \in M(n)} E_{mn}^{(i)} \quad (3\text{-}4)$$

3.2.4 2D-normalized MS shuffled decoding

2D-normalized MS shuffled decoding 是將 MS shuffled decoding 的 chcek node 乘上一個正規化係數 β_1，對 MS shuffled decoding 的 bit node 乘上另一個 β_2 正規化係數，用兩個補償係數分別修正 check node 與 bit node，請看 3-5 式。

$$E_{mn}^{(i)} = [\prod_{n' \in N(m)\backslash n} \mathrm{sgn}(Z_{mn'}^{(i)or(i-1)})].[\min\{|Z_{mn'}|^{(i)or(i-1)}_{n' \in N(m)\backslash n} .\beta_1\}]$$

$$Z_{mn}^{(i)} = F_n + \sum_{m' \in M(n)\backslash m} \{|E_{m'n}|^{(i)} .\beta_2\} \;\; ; \;\; Z_n^{(i)} = F_n + \sum_{m \in M(n)} E_{mn}^{(i)} \qquad (3\text{-}5)$$

3.2.5 static compensated MS shuffled decoding 與 dynamic compensated MS shufffled decoding

補償系數又分成使用固定補償係數的靜態補償與使用不同補償係數的動態補償。

動態補償，有兩種不同補償方式，第一種是對 MS shuffled decoding 的 check node update 最小項與次小項做補償[23]，第二種為對不同的 SNR 給予不同補償係數。本論文採用第二種方式做動態補償。

所以，3.2.1~3.2.4 節所提到的四種補償演算法，使用靜態補償係數，為 static compensated MS shuffled decoding，使用動態補償係數，則稱為 dynamic compensated MS shuffled decoding。

第四章　模擬結果

我們使用 Standard 802.11n 的同位檢查矩陣(如附件 1)，分別對 codeword 長度為 648、1296、1944，code rate 為 1/2、2/3、3/4、5/6 的 compensated MS shuffled decoding、MS shuffled decoding 與 shuffled BP decoding 三種演算法做模擬。圖 20 為 compensated MS shuffled decoding 的補償係數求法。

經由模擬，觀察 compensated MS shuffled decoding 的解碼效能是否能提升到接近 shuffled BP decoding。模擬結果發現，對於 Standard 802.11n LDPC code 來說，大部份的 compensated MS shuffled decoding 的 BER-SNR 曲線十分接近 shuffled BP decoding 的 BER-SNR 曲線，顯示補償的確發揮很好的效果。

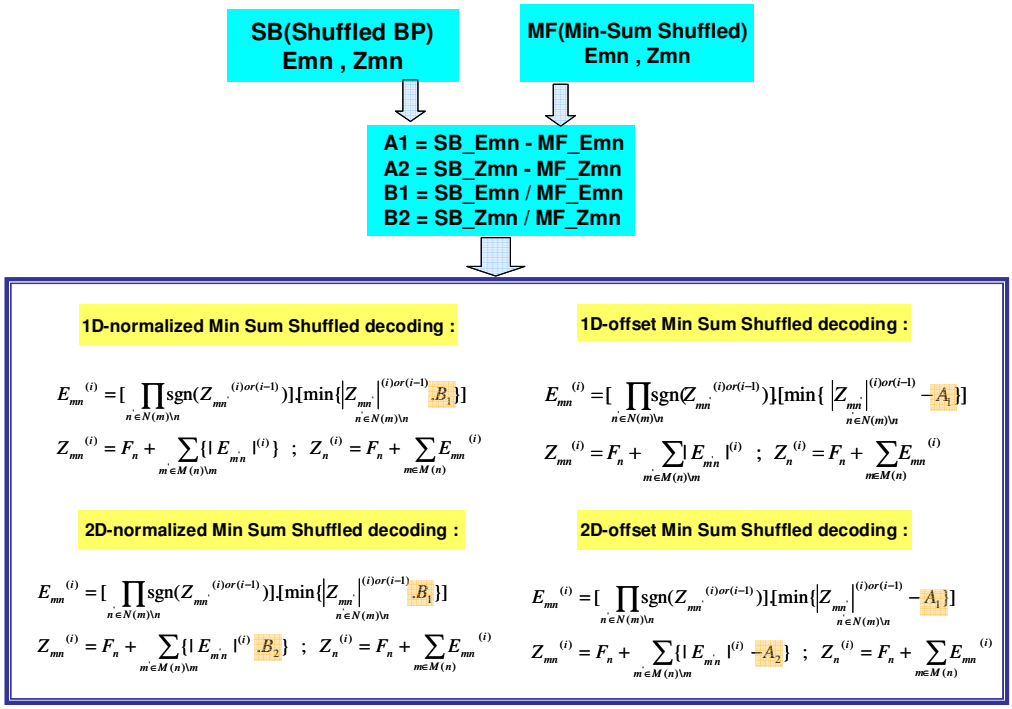


圖 20　compensated MS shuffled decoding

4.1　靜態補償的模擬結果

為了簡化分析，我們對 MS shuffled decoding 使用靜態補償係數做補償。圖 21、22，表示碼字長度分別為 648 與 1296，code rate 為 1/2、2/3、3/4、5/6，使用固定補償係數的 static 1D- normalized MS shuffled decoding，static 1D-offset MS shuffled decoding，static 2D-normalized MS shuffled decoding，static 2D-offset MS shuffled decoding、MS Shufled decoding 以及 shuffled BP decoding 的解碼錯誤效能。圖下方標示出各種靜態補償由好到壞的效能評比，並將所使用的固定補償係數數值標示在圖中的註解欄。

從模擬結果我們可以看出，static compensated MS shuffled decoding 的 BER 比 MS shuffled decoding 小很多，而且大部份的結果顯示 static compensated MS shuffled decoding 的 BER-SNR 曲線很接近 shuffled BP decoding 的 BER-SNR 曲線 ，代表補償效果十分良好。

補償效果最顯著的，為 codeword 為 648，code rate 為 2/3 這組模擬，請看圖 21。static 2D-normalizrd MS shuffled decoding 甚至與 shuffled BP decoding 的 BER-SNR 曲線重合在一起。透過觀察，我們發現 static compensated MS shuffled decoding 由好到壞的解碼效能次序，大致如下，static 2D-normalized ＞ static 2D-offset ＞ static 1D-offset ＞ static 1D-normalized。

但是有些情行，如圖 21，codeword=648，code rate 為 5/6 時，compensated MS shuffled decoding 不管用哪種補償係數來補償，解碼結果都差不多，這時候，可以選用 1D-offset 來做補償，硬體較簡單。所以，我們可以視實際使用情行，來選擇合適的補償係數。

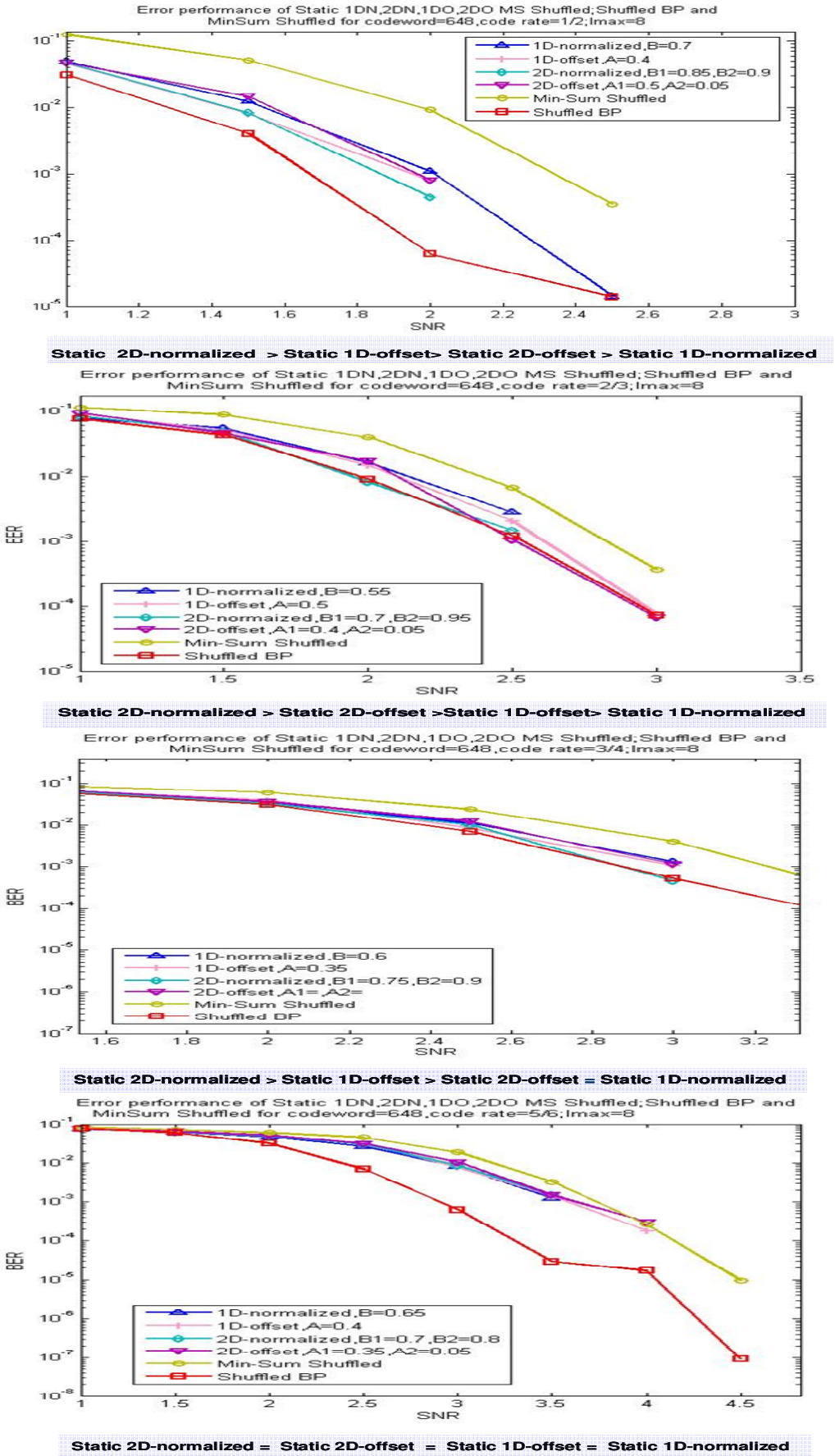
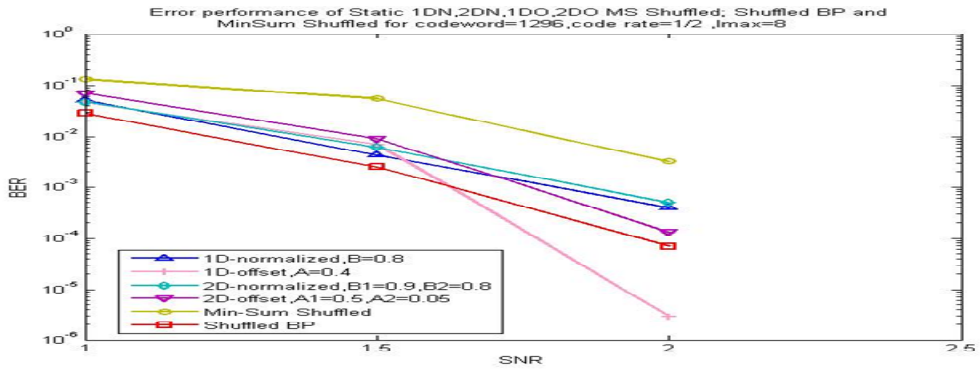
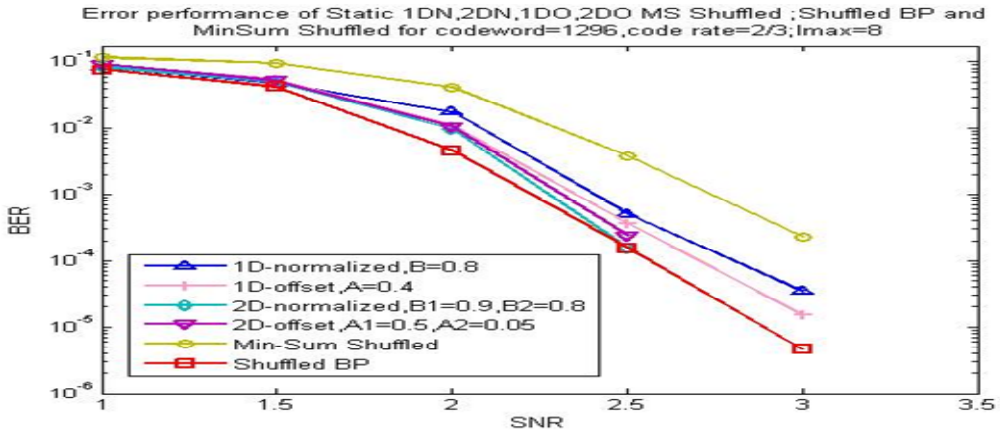


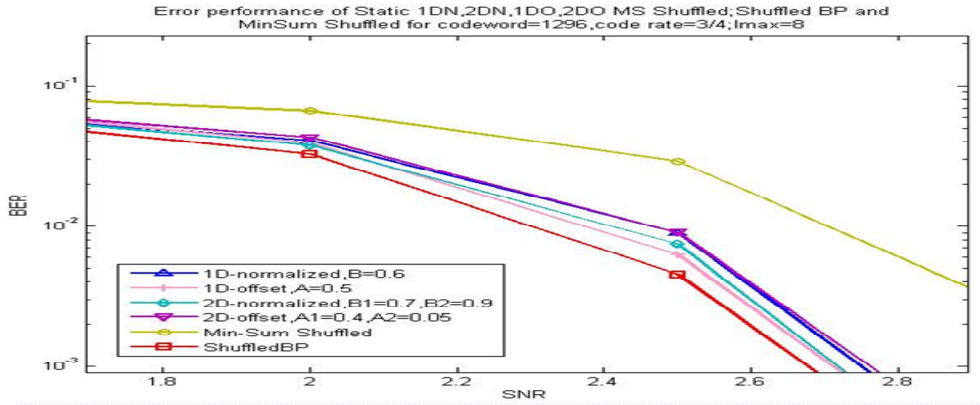
圖 21　codeword=648 之 static compensated MS shuffled decoding 模擬結果



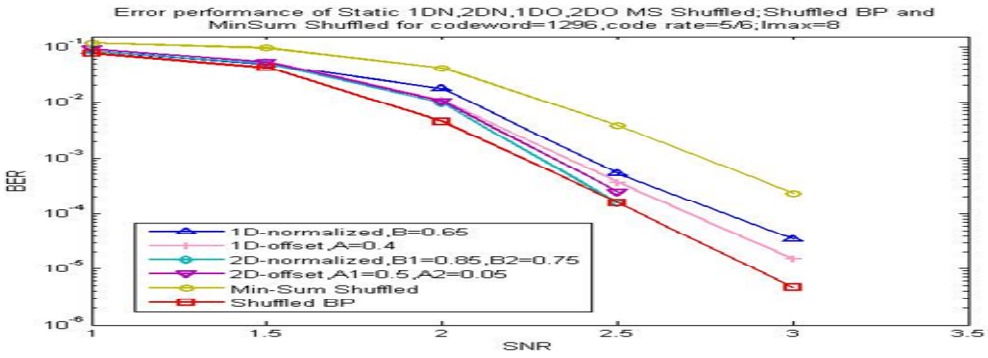
Static 1D-normalized > Static 1D-offset > Static 2D-normalized >Static 2D-offset



Static 2D-normalized=Static 2D-offset > Static 1D-offset > Static 1D-normalized



Static 1D-offset>Static 2D-normalized >Static 2D-offset=Static 1D-normalized



Static 2D-normalized =Static 2D-offset > Static 1D-offset > Static 1D-normalized

圖 22　codeword=1296 之 static compensated MS shuffled decoding 模擬結果

4.2 動態補償的模擬結果

由於補償係數與 SNR(平均訊號功率相對平均雜訊功率的比值)有關，所以，我們也針對不同得的 SNR 值採用不同的補償係數，對 MS shuffled decoding 做動態補償。在這裡，我們以 dynamic 1D-normalized MS shuffled decoding 做為動態補償的研究。

圖 23~25 代表，codeword 為 648、1296、1944，code rate 為 1/2、2/3、3/4、5/6，最大遞迴解碼次數 Imax 為 8 的 dynamic 1D-normalized MS shuffled decoding 與 MS shuffled decoding 以及 shuffled BP decoding 三種解碼演算法的 error performance 模擬結果。

從圖 23~25 中，我們看到大部份結果顯示 dynamic 1D-normalized MS shuffled decoding 的 BER(位元錯誤錯誤率)都比 MS shuffled decoding 的 BER 小很多，很接近 shuffled BP decoding 的 BER-SNR 曲線，表示補償確實有其效用。補償效果最好的情行，如圖 25，碼長為 1944，code rate 為 5/6 時，效果尤其顯著，dynamic 1D-normalized MS shuffled decoding 的 BER-SNR 曲線甚至與 shuffled BP decoding 的 BER-SNR 曲線重合。

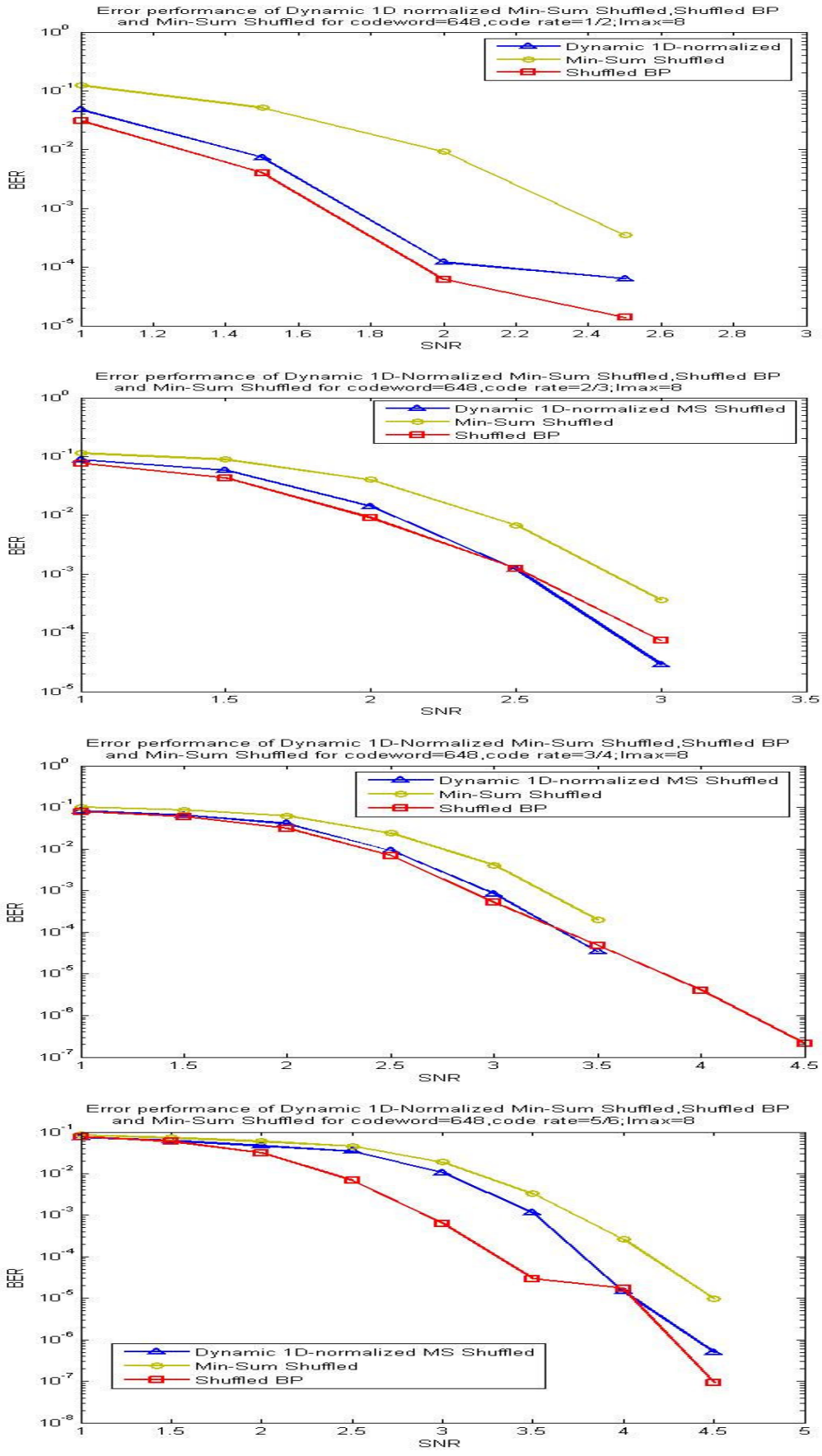
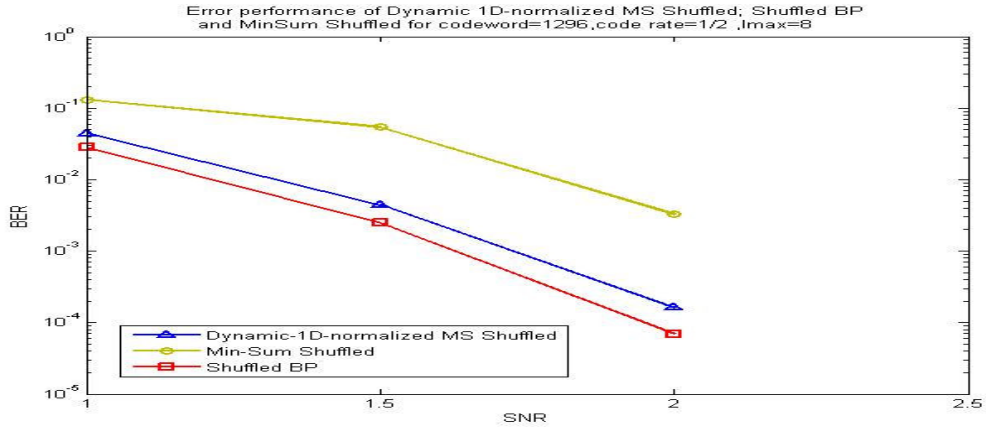
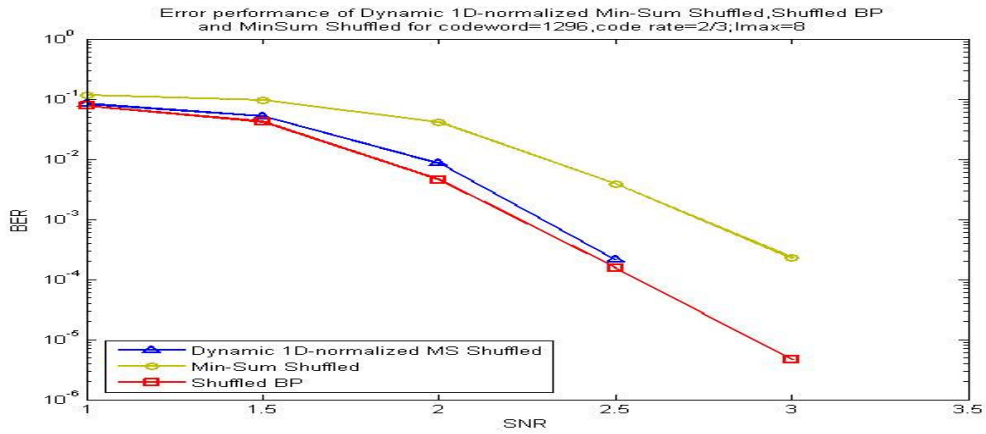
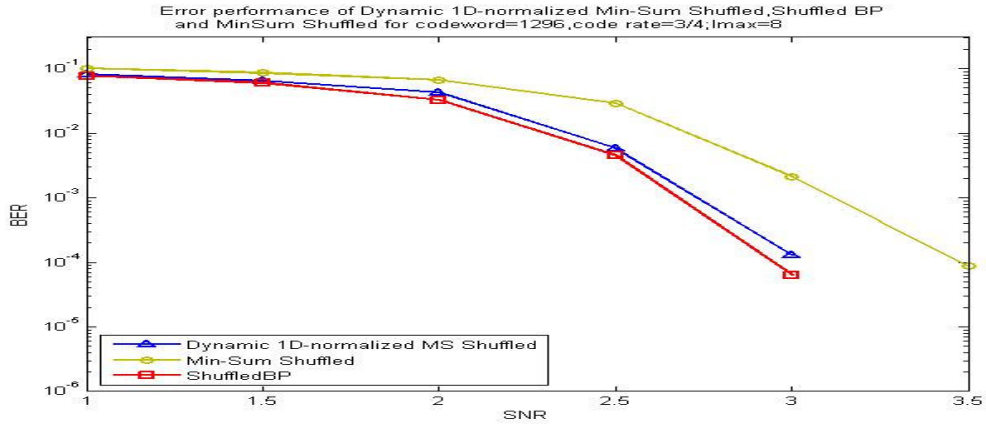


圖 23　codeword=648 之 dynamic compensated MS shuffled decoding 模擬結果







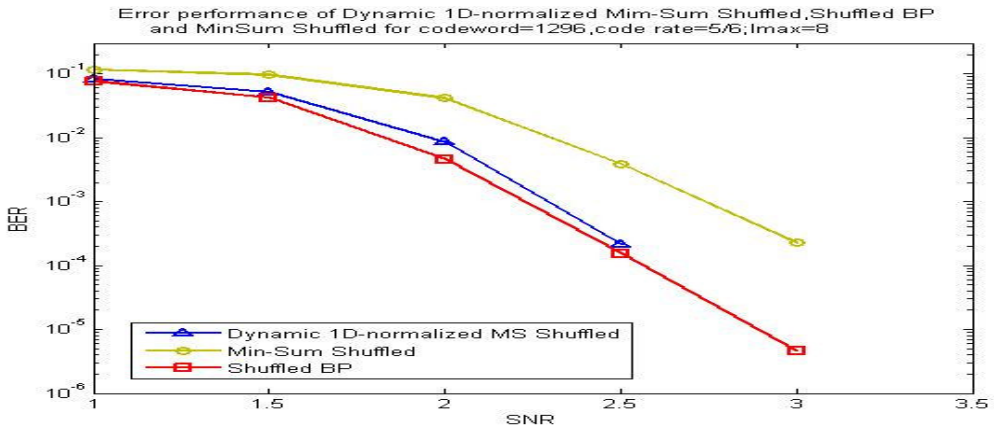


圖 24 codeword=1296 之 dynamic compensated MS shuffled decoding 模擬結果

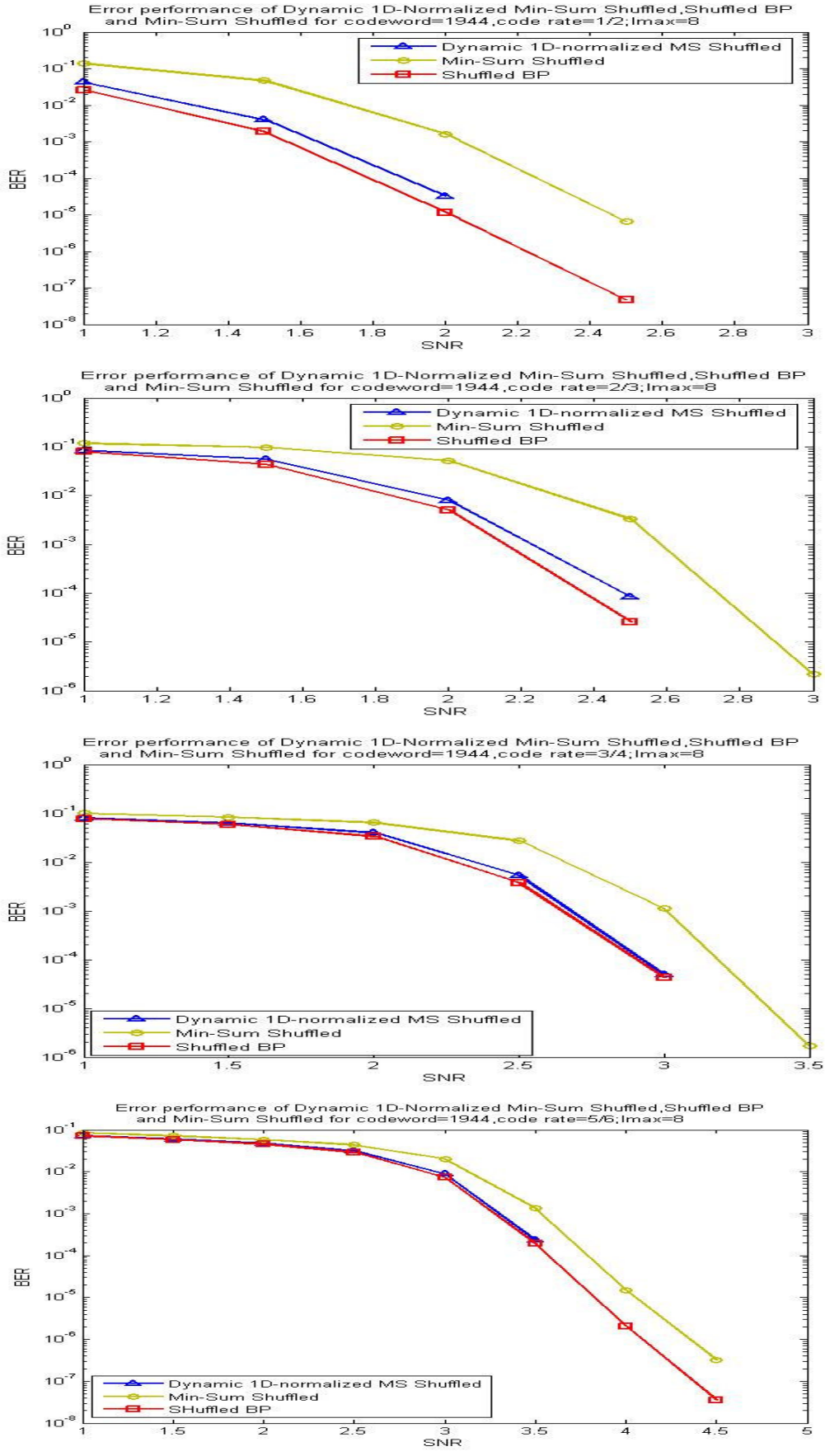


圖 25 codeword=1944 之 dynamic compensated MS shuffled decoding 模擬結果

4.2.1 動態補償係數之模擬數據分析

我們將 4.2 節模擬所得到的動態 1D-normalized 補償係數，做成動態 1D-normalized 補償係數與 SNR 值的關係圖，如圖 26~圖 28 所示。

圖 26~28 分別表示碼長為 648、1296、1944 與 code rate 為 1/2、2/3、3/4、5/6 的 SNR 與 1D normalized 補償係數 Beta 的關係。我們觀察到，code rate 愈大，1D normalized 係數對 SNR 的斜率愈大，反之亦然。所以，當 code rate 較小的時候(比如 code rate=1/2)，表示 1D normalized 係數對 SNR 值的變化較不敏感，所以可以取動態 1D normalized 係數平均值做靜態 1D-normalized 補償的靜態補償係數。

另外，從圖中可以觀察出，code rate 小的碼，它的 1D normalized 補償係數會比 code rate 大的碼的 1D normalized 補償係數來得大，圖內註解框內註記的是 1D normalized 補償係數平均值(簡稱 mean B)。

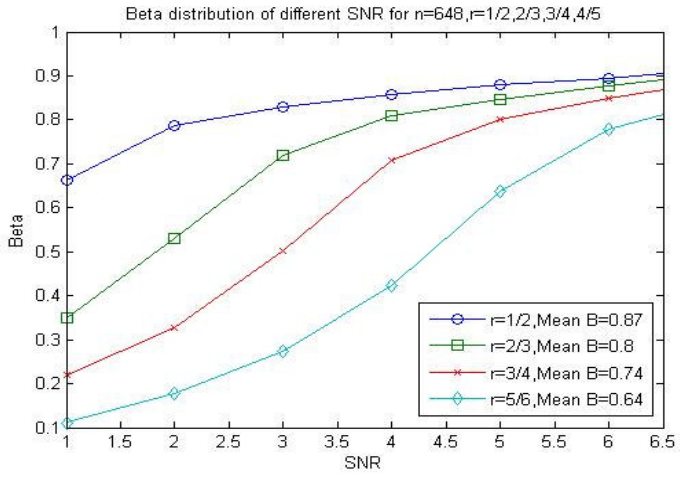


圖 26　codeword=648 之動態補償係數與 SNR 之關係圖

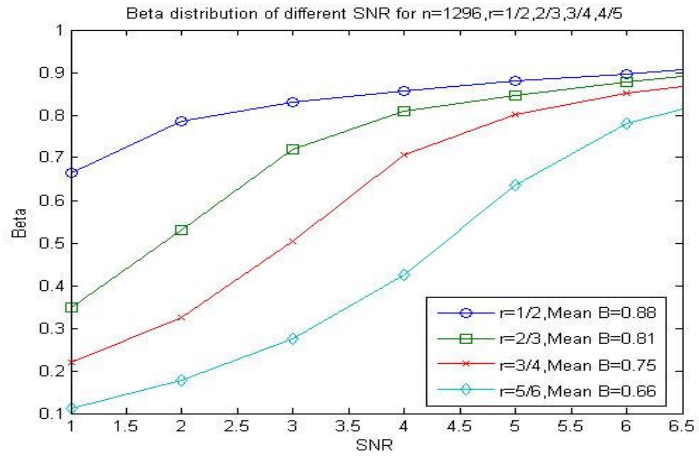


圖 27　codeword=1296 之動態補償係數與 SNR 之關係圖

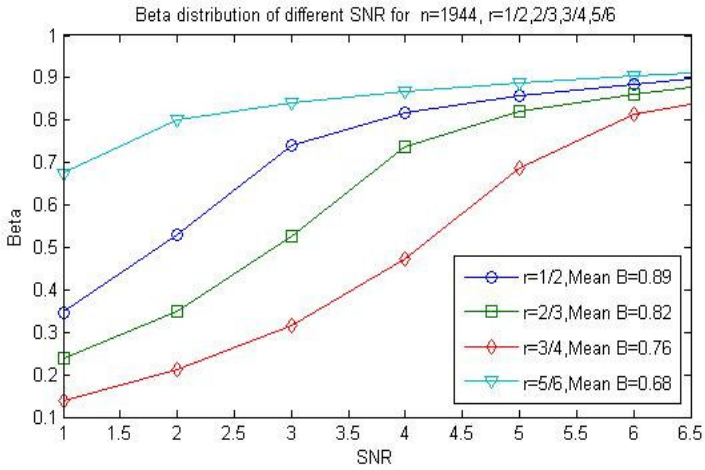


圖 28　codeword=1944 之動態補償係數與 SNR 之關係圖

4.3 靜態補償與動態補償的解碼效能比較

我們將四種靜態補償 static 1D-normalized MS shuffled decoding、static 1D-offset MS shuffled decoding、static 2D-normalized MS shuffled decoding、static 2D-offset MS shuffled decoding 與動態補償 dynamic1D-normalized MS shuffled decoding 的解碼效能結果與 MS shuffled decoding、shuffled BP decoding 做比較，如圖 60~67 所示，觀察靜態補償與動態補償的解碼效能。

圖 29~32 分別是 codeword 為 648、1296，code rate 為 1/2、2/3、3/4、5/6 的 dynamic 1D-normalized MS shuffled decoding，static compensated MS shuffled decoding(1D-normalized、1D-offaet、2D-normalized、2D-offset)，MS shuffled decoding 以及 shuffled BP decoding 的解碼效能比較，圖的下方標示了各種補償演算法由好到壞的效能評比。

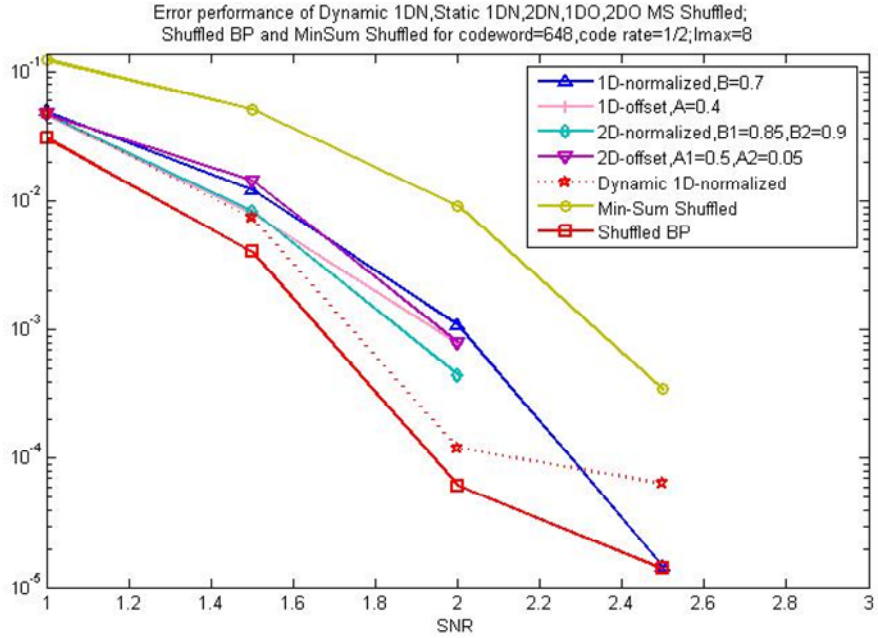
經過我們從圖 29~32 的觀察與統計，codeword=648、1296，code rate=1/2、2/3、3/4、5/6 的補償的效能大致如下:

static 2D-normalized ≧ dynamic 1D-normalized > static 2D-offset > static 1D-offset > static 1D-normalized

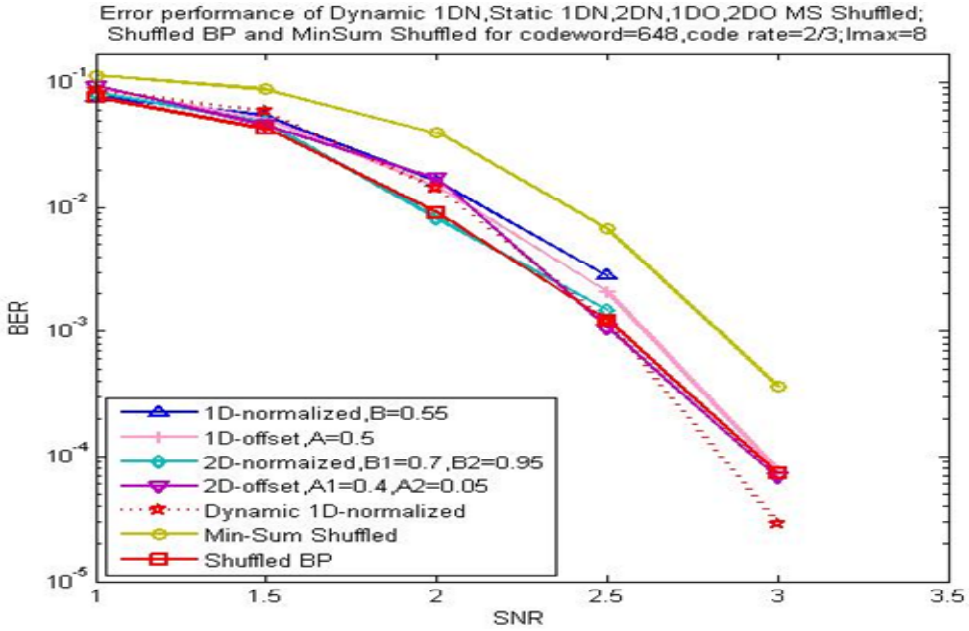
從以上的縱合比較，我們發現 2D 的補償比 1D 來得好，而最差的是靜態的 1D-normalized，但是 1D-normalized 若用動態來補償，卻可以從最差的 error performance 躍升成最好的,可見動態補償仍然有其效果。

除此之外，我們可以依照自己的須求，去選則擇合適的補償系數，比如 codeword 為 1296，code rate 為 1/2 這組模擬(圖 31)，如果須求是 SNR 大於 1.5db

時想要有更低的 error performance，則 1D-offset 就比其他的補償有更低的位元錯誤率(BER)。

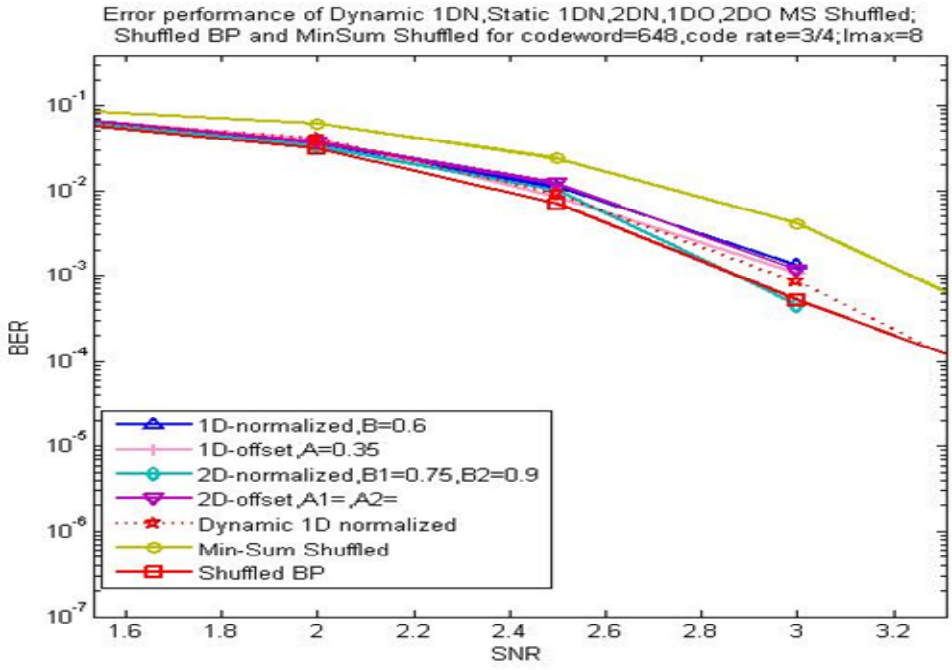


Dynamic 1D-normalized > Static 2D-normalized
> Static 1D-offset> Static 2D-offset > Static 1D-normalized



Static 2D-normalized > Dynamic 1D-normalized
> Static 2D-offset >Static 1D-offset> Dynamic 1D-normalized

圖 29　codeword=648，coderate=1/2、2/3 之 compensated MS shuffled decoding 模擬結果



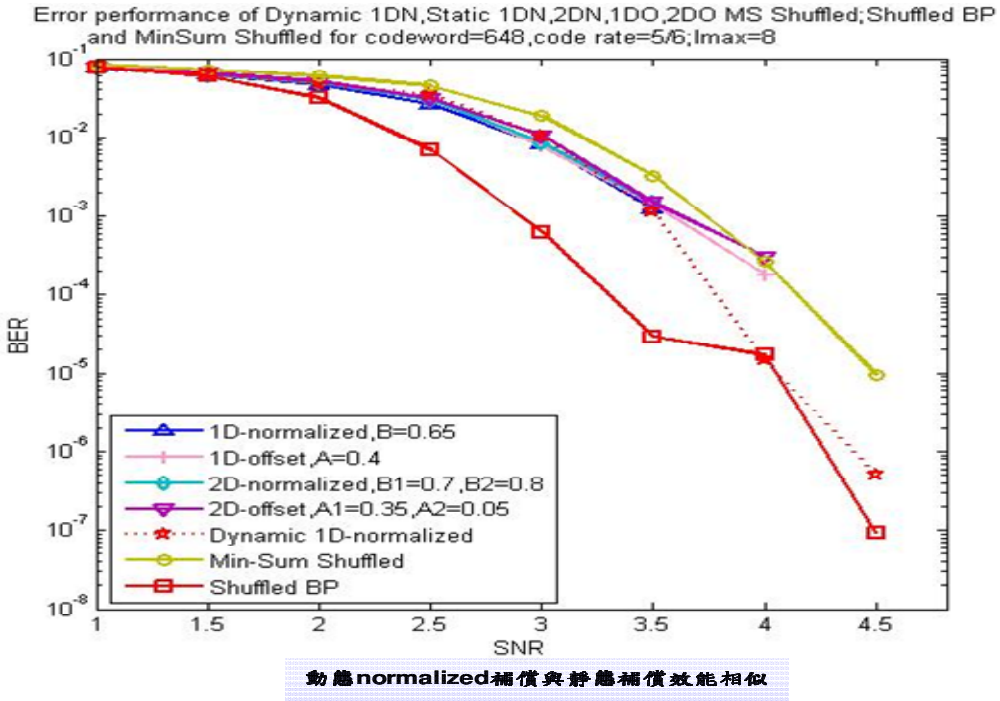
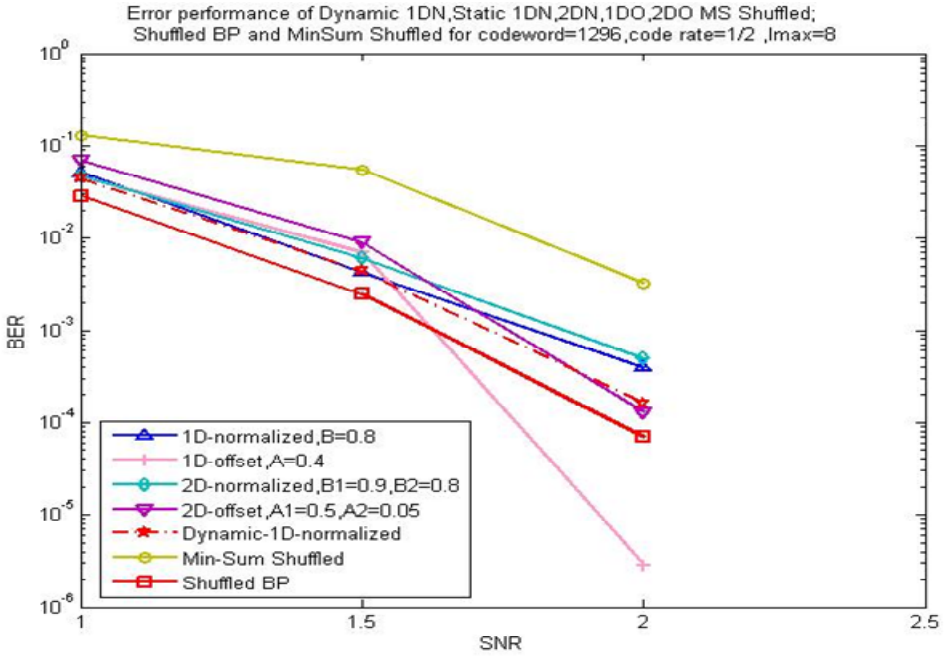
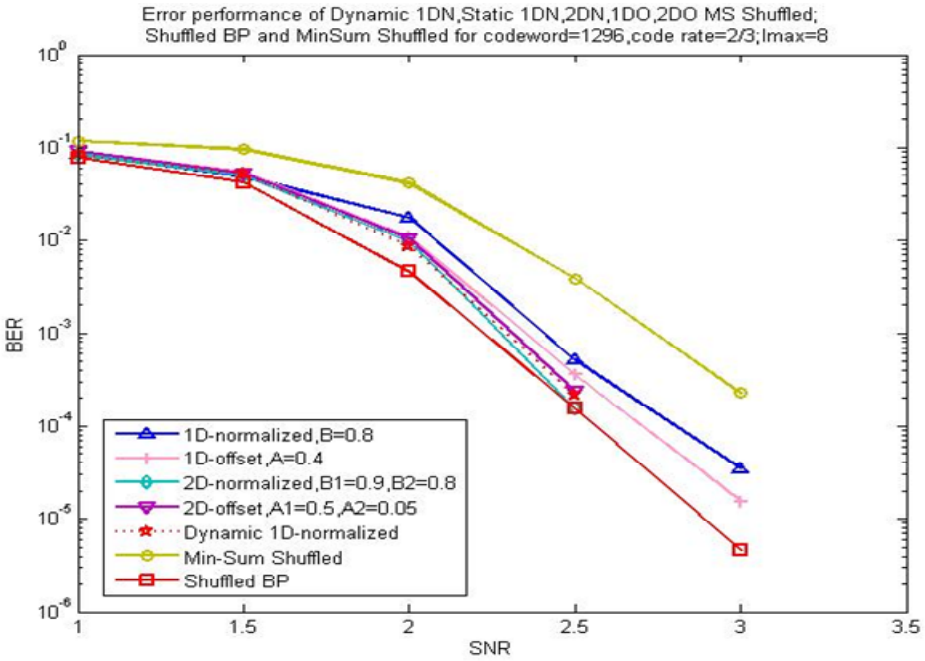


圖 30　codeword=648，coderate=3/4、5/6 之 compensated MS shuffled decoding 模擬結果



Dynamic normalized > Static 1D-normalized
> Static 2D-normalized > Static 1D-offset >Static 2D-offset



Static 2D-normalized >Dynamic normalized
=Static 2D-offset > Static 1D-offset > Static 1D-normalized

圖 31　codeword=1296，coderate=1/2、2/3 之 compensated MS shuffled decoding 模擬結果

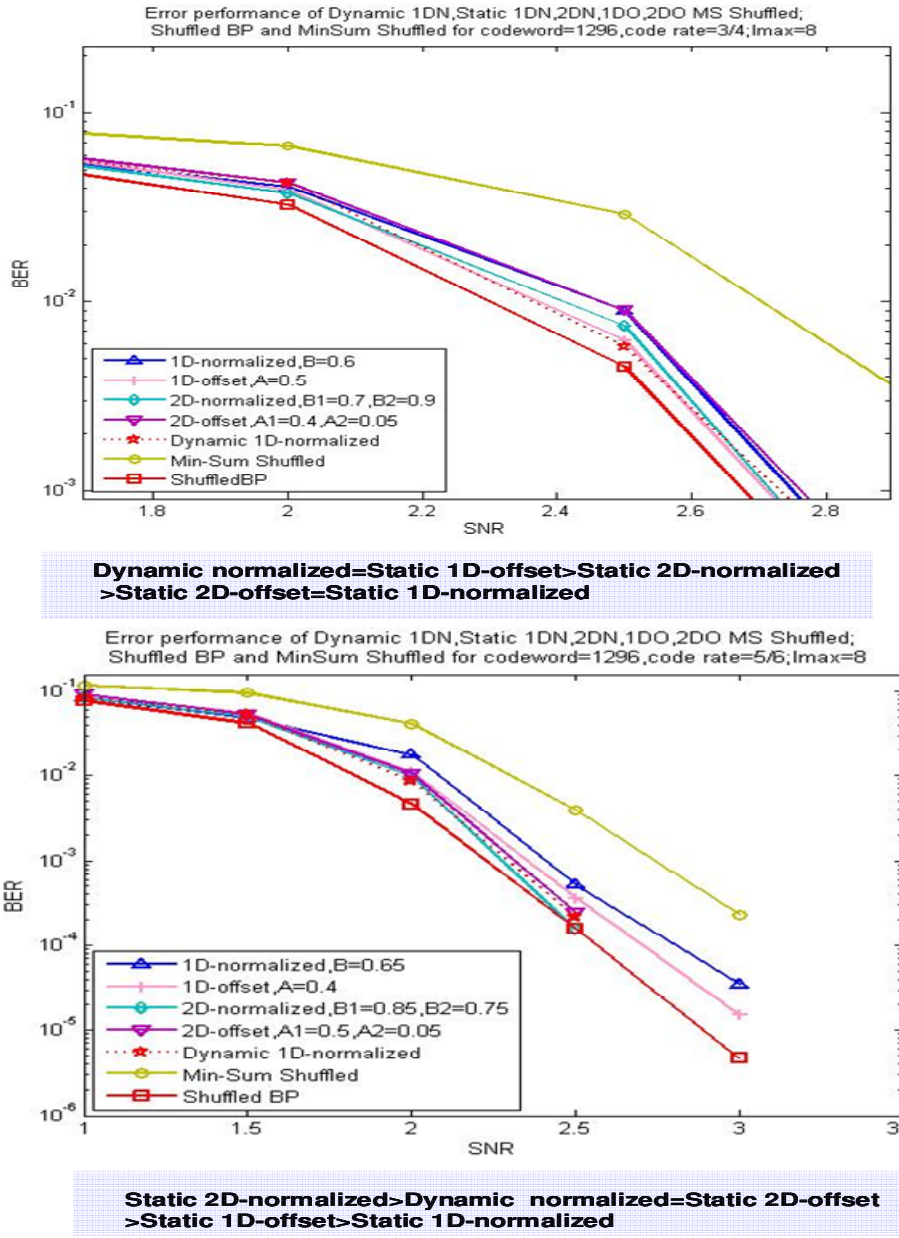


圖 32　codeword=1296，coderate=3/4、5/6 之 compensated MS shuffled decoding 模擬結果

4.4　補償係數模擬數據

我們將靜態與動態補償模擬結果，所得到的補償係數列在表格 1~5。

表格 1、2 為 static compensated MS shuffled decoding(1D-normalized、1D-offaet、

2D-normalized、2D-offset)模擬所得到的靜態補償係數。表格 1、2 各代表碼長為 648、1296 在 SNR 為 1~6，R(code rate)為 1/2、2/3、3/4、5/6 的靜態 1D-normalized、1D-offaet、2D-normalized、2D-offset 補償係數。

表格 3~5 為 dynamic 1D-normalized MS shuffled decoding 模擬所得到的動態 1D-normalized 補償係數，表格 3、4、5 各代表碼長為 648、1296、1944 在 SNR 為 1~6，R(code rate)為 1/2、2/3、3/4、5/6 的動態 1-D normalized 補償係數。

表格 1　codeword=648，code rate=1/2、2/3、3/4、5/6 的靜態補償係數

Codeword =648	Static Factor					
	1D-normalized	1D-Offset	2D-normalized		2D-offset	
	β	α	$\beta 1$	$\beta 2$	$\alpha 1$	$\alpha 2$
R=1/2	0.7	0.4	0.85	0.9	0.5	0.05
R=2/3	0.55	0.5	0.7	0.95	0.4	0.05
R=3/4	0.6	0.35	0.75	0.9	0.5	0.05
R=5/6	0.65	0.4	0.7	0.8	0.35	0.05

表格 2　codeword=1296，code rate=1/2、2/3、3/4、5/6 的靜態補償係數

Codeword =1296	Static Factor					
	1D-normalized	1D-Offset	2D-normalized		2D-offset	
	β	α	$\beta 1$	$\beta 2$	$\alpha 1$	$\alpha 2$
R=1/2	0.8	0.4	0.9	0.8	0.5	0.1
R=2/3	0.6	0.5	0.8	0.8	0.5	0.05
R=3/4	0.6	0.5	0.7	0.9	0.4	0.05
R=5/6	0.65	0.4	0.85	0.75	0.5	0.05

表格 3　codeword=648，code rate=1/2、2/3、3/4、5/6 的動態補償驗係數

codeword =648	Dynamic 1D-normalized Factor (β)											
	SNR											
	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5
R=1/2	0.65	0.76	0.81	0.84	0.87	0.89	0.91	0.92	0.94	0.94	0.95	0.96
R=2/3	0.34	0.51	0.67	0.77	0.83	0.86	0.89	0.9	0.93	0.94	0.96	0.97
R=3/4	0.22	0.33	0.49	0.65	0.77	0.83	0.87	0.89	0.92	0.94	0.96	0.97
R=5/6	0.99	0.16	0.26	0.4	0.59	0.73	0.82	0.86	0.9	0.93	0.95	0.96

表格 4　codeword=1296，code rate=1/2、2/3、3/4、5/6 的動態補償係數

codeword =1296	Dynamic 1D-normalized Factor (β)											
	SNR											
	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5
R=1/2	0.66	0.79	0.83	0.86	0.88	0.9	0.91	0.93	0.94	0.95	0.96	0.96
R=2/3	0.35	0.53	0.72	0.81	0.85	0.88	0.9	0.92	0.93	0.95	0.96	0.97
R=3/4	0.22	0.33	0.5	0.71	0.8	0.85	0.88	0.91	0.92	0.94	0.96	0.97
R=5/6	0.11	0.18	0.27	0.42	0.64	0.78	0.85	0.89	0.91	0.93	0.95	0.96

表格 5　codeword=1944，code rate=1/2、2/3、3/4、5/6 的動態補償係數

codeword =1944	Dynamic 1D-normalized Factor (β)											
	SNR											
	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5
R=1/2	0.68	0.8	0.84	0.87	0.89	0.9	0.92	0.94	0.95	0.96	0.96	0.97
R=2/3	0.35	0.53	0.74	0.82	0.86	0.88	0.91	0.93	0.94	0.95	0.96	0.97
R=3/4	0.24	0.35	0.53	0.74	0.82	0.86	0.9	0.92	0.93	0.94	0.96	0.97
R=5/6	0.14	0.21	0.31	0.47	0.69	0.81	0.86	0.9	0.92	0.94	0.95	0.97

第五章 Summary

compensated MS shuffled decoding 主要是修正 MS shuffled decoding 因為簡化硬體所造成解碼效能下降，所以補償的結果，以愈接近 shuffled BP decoding 為目標。我們從第 4 章的模擬結果，可以看出不管是使用靜態補償係數或是動態補償係數，結果都比 MS shuffled decoding 的 BER(bit error rate)下降很多，有些情形，compensated MS shuffled decoding 的 BER-SNR 曲線甚至與 shuffled BP decoding 的重合（比如 codeword 為 648，code rate 為 2/3 的 static 2D-normalized MS shuffled decoding 與 codeword 為 1944，code rate 為 5/6 的 dynamic 1D-normalized MS shuffled decoding），表示補償確實發揮很好的修正效果。

從模擬結果，我們觀察到，compensated MS shuffled decoding 的補償係數由好到壞的解碼效能次序，大致如下，static 2-D normalized ≧ dynamic 1-D normalized ＞ static 2-D offset ＞ static1-D offset ＞ static 1-D normalized。

由於動態補償在不同 SNR 值有不同的補償係數，較複雜。所以，若是動態與靜態補償的解碼效果相似時，建議以靜態係數補償為宜，選擇上以 1D 優於 2D，offset 優於 normalized 為原則。

由於本論文的研究以 IEEE Standard 802.11n 的同位檢查矩陣進行模擬，在未來的研究方向上，希望能納入其他 Standard，比如 IEEE Standard 802.16，或其他的 LDPC codes 來做模擬、分析等研究比較，觀察 compensated MS shuffled decoding 在別種 code 上的使用效果，並分析補償係數與同位檢查矩陣的相關性，藉由理論推導與實驗模擬找出應用於 MS shuffled decoding 的最佳補償係數。

參考文獻

[1] Bernard Sklar,“Digital Communications Fundamentals and Applications Second EditionCommunication Engineering Services”,Tarzana California and University of California,Los Angeles

[2] R.G.Gallager,“Low Density Parity Check Codes”,IRE Trans.Inform.Theory,vol. IT-8,pp.21-28,Jan.1962.

[3] D.J.C.MacKay and R. M. Neal,“Near Shannon lomit performance of low density parity check codes”,Electronics Letters,vol.33,no.6,pp.457-458,Mar.1997.

[4] D.J.C.MackKay,“Good error-correcting codes based on very sparse matrices”, IEEE Trans.Inform.Theory,vol.45,no.2,pp.399-431,Mar.1999.

[5] F.Guiloud,“Generic architechture for LDPC codes decoding”,Ph.D.dissertation, ENST Paris,Paris,France,2004.

[6] T.J.Richardson and R.L.Urbanke,“Efficient encodingof low density parity check codes”,IEEE Trans.Inform.Theory,vol.47,no.2,pp.638-656,Feb.2001.

[7] IEEE Std 802.16e-2005,“Air Interface for Fixed and Mobile Broadband Wireless Access Systems”.

[8] Shih-Hsien Liu, “Design and Implementation of a Configurable LDPC Decoder for IEEE 802.16e and 802.11n”, NCTU Master Thesis, 2007.

[9] J.Chen,A.Dholakia,E. Eleftheriou,M.P.C.Fossorier,and X.Y. Hu,“Reduced-compl

exity decoding of ldpc codes",IEEE.Trans.Commun,vol.53,pp.1288-1299,Aug.2005

[10] Kuan-Jung Huang,"Detection and LDPC Codes Decoding for MIMO-OFDM System",NCTU Master Thesis,2006.

[11] M.P.C.Fossorier,M.Mihaljevic,and H.Imai,"Reduced complexity iterative decoding of low density parity check codes based on belief propagation",IEEE Trans.Commun.,vol.47,no.5,pp.673-680,May.1999.

[12] Yong-Jhe Lin,"A Atudy on LDPC Coding Technique and Its Application Digital Video Broadcasting System",NCTU Master Thesis,2005.

[13] J.Chen and M.P.C.Fossorier,"Density evolution for two improved BP-based decoding algorithm of LDPC codes",IEEE Comm.Lett,vol6,no.5,pp.208-210, May 2002.

[14] J.Chen and M.P.C.Fossorier,"Near Optimum Universal Belief Propagation Based Decoding of Low-DensityParity Check Codes",IEEE Trans. Commun, vol.50,no.3,pp.406-414,March 2002.

[15] J.Zhang,M.Fossorier,D.Gu,andJ.Zhang,"Improved min-sum decoding of ldpc codes using 2-dimensional normalization",in IEEE GLOBECOM'05,vol.3,pp.1187-1192,Nov.2005.

[16] J.Zhang,M.Fossorier,D.Gu,and J.Y.Zhang,"Two-Dimensional Correction for Min-Sum Decoding of irregular LDPC codes",submitted to IEEE Commun. Lett.

[17] J.Zhang,M.Fossorier,D.Gu,and J.Zhang,“Improved min-sum decoding of ldpc codes using 2-dimensional normalized”,in IEEE GLOBE COM’05,vol.3,pp. 1187–1192,Nov.2005.

[18] J.Zhang and M.Fossorier,“Shuffled belief propagation decoding”,in Proc. 36t h Annu. Asilomar Conf. Signals,Syst.,Computers,pp.8-15.Nov.2002.

[19] J.Zhang,M.P.C.Fossorier,“shuffled Iterative Decoding”,IEEE Trans. Commun. vol.53,no.2,pp.209-213,Feb.2005.

[20] H.Kfir and I.Kanter,“Parallel versus sequential updating for belief propagati on decoding”,Phys.Rev.E,submitted for publication.

[21] Shihyao Wang,“Belief Propagation Based Decoding Schedules for Low Den sity Parity Check Codes and their Behavior Analysis”,NCTU Mast Thesis, 2007.

[22] Chin-Hung Chen,“Trend and Techniques in Wireless SoC”,SoC Technical J ournal pp.152-165.

[23] Y.C.Liao,C.C.Lin,C.W.Liu,andH.C.Chang,“Self-Compensation Technique for S implified Belief-Propagation Algorithm”,in IEEE Transactions on Signal Pr ocessing,vol.55,no.6,Jun.2007.

附件1　802.11n之Parity Check Matrix

Table n103
Matrix prototypes of parity-check matrices for codeword block length *n*= 648 bits.
Subblock size is *Z*= 27 bits.

(a)
Code rate *R*= 1/2.

0	-	-	-	0	0	-	-	0	-	-	0	1	0	-	-	-	-	-	-	-	-	-	-	-
22	0	-	-	17	-	0	0	12	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-	-
6	-	0	-	10	-	-	-	24	-	0	-	-	-	0	0	-	-	-	-	-	-	-	-	-
2	-	-	0	20	-	-	-	25	0	-	-	-	-	-	0	0	-	-	-	-	-	-	-	-
23	-	-	-	3	-	-	-	0	-	9	11	-	-	-	-	0	0	-	-	-	-	-	-	-
24	-	23	1	17	-	3	-	10	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-	-
25	-	-	-	8	-	-	-	7	18	-	-	0	-	-	-	-	-	0	0	-	-	-	-	-
13	24	-	-	0	-	8	-	6	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-
7	20	-	16	22	10	-	-	23	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-
11	-	-	-	19	-	-	-	13	-	3	17	-	-	-	-	-	-	-	-	-	0	0	-	-
25	-	8	-	23	18	-	14	9	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-
3	-	-	-	16	-	-	2	25	5	-	-	1	-	-	-	-	-	-	-	-	-	-	0	0

(b)

Code rate *R*= 2/3.

25	26	14	-	20	-	2	-	4	-	-	8	-	16	-	18	1	0	-	-	-	-	-	-	-
10	9	15	11	-	0	-	1	-	-	18	-	8	-	10	-	-	0	0	-	-	-	-	-	-
16	2	20	26	21	-	6	-	1	26	-	7	-	-	-	-	-	-	0	0	-	-	-	-	-
10	13	5	0	-	3	-	7	-	-	26	-	-	13	-	16	-	-	-	0	0	-	-	-	-
23	14	24	-	12	-	19	-	17	-	-	-	20	-	21	-	0	-	-	-	0	0	-	-	-
6	22	9	20	-	25	-	17	-	8	-	14	-	18	-	-	-	-	-	-	-	-	0	0	-
14	23	21	11	20	-	24	-	18	-	19	-	-	-	-	22	-	-	-	-	-	-	-	0	0
17	11	11	20	-	21	-	26	-	3	-	-	18	-	26	-	1	-	-	-	-	-	-	-	0

(c)

Code rate *R*= 3/4.

16	17	22	24	9	3	14	-	4	2	7	-	26	-	2	-	21	-	1	0	-	-	-	-	-
25	12	12	3	3	26	6	21	-	15	22	-	15	-	4	-	-	16	-	0	0	-	-	-	-
25	18	26	16	22	23	9	-	0	-	4	-	4	-	8	23	11	-	-	-	0	0	-	-	-
9	7	0	1	17	-	-	7	3	-	3	23	-	16	-	-	21	-	0	-	-	0	0	-	-
24	5	26	7	1	-	-	15	24	15	-	8	-	13	-	13	-	11	-	-	-	-	-	0	0
2	2	19	14	24	1	15	19	-	21	-	2	-	24	-	3	-	2	1	-	-	-	-	-	0

(d)

Code rate R= 5/6.

17	13	8	21	9	3	18	12	10	0	4	15	19	2	5	10	26	19	13	13	1	0	-	-
3	12	11	14	11	25	5	18	0	9	2	26	26	10	24	7	14	20	4	2	-	0	0	-
22	16	4	3	10	21	12	5	21	14	19	5	-	8	5	18	11	5	5	15	0	-	0	0
7	7	14	14	4	16	16	24	24	10	1	7	15	6	10	26	8	18	21	14	1	-	-	0

Table n104
Matrix prototypes of parity-check matrices for codeword block length n= 1296 bits.
Subblock size is Z= 54 bits.

(a)

Code rate R= 1/2.

40	-	-	-	22	-	49	23	43	-	-	-	1	0	-	-	-	-	-	-	-	-	-	-
50	1	-	-	48	35	-	-	13	-	30	-	-	0	0	-	-	-	-	-	-	-	-	-
39	50	-	-	4	-	2	-	-	-	-	49	-	-	0	0	-	-	-	-	-	-	-	-
33	-	-	38	37	-	-	4	1	-	-	-	-	-	-	0	0	-	-	-	-	-	-	-
45	-	-	-	0	22	-	-	20	42	-	-	-	-	-	-	0	0	-	-	-	-	-	-
51	-	-	48	35	-	-	-	44	-	18	-	-	-	-	-	-	0	0	-	-	-	-	-
47	11	-	-	-	17	-	-	51	-	-	-	0	-	-	-	-	-	0	0	-	-	-	-
5	-	25	-	6	-	45	-	13	40	-	-	-	-	-	-	-	-	-	0	0	-	-	-
33	-	-	34	24	-	-	-	23	-	-	46	-	-	-	-	-	-	-	-	0	0	-	-
1	-	27	-	1	-	-	-	38	-	44	-	-	-	-	-	-	-	-	-	-	0	0	-
-	18	-	-	23	-	-	8	0	35	-	-	-	-	-	-	-	-	-	-	-	-	0	0
49	-	17	-	30	-	-	-	34	-	-	19	1	-	-	-	-	-	-	-	-	-	-	0

(b)

Code rate R= 2/3.

39	31	22	43	-	40	4	-	11	-	-	50	-	-	-	6	1	0	-	-	-	-	-	-
25	52	41	2	6	-	14	-	34	-	-	-	24	-	37	-	-	0	0	-	-	-	-	-
43	31	29	0	21	-	28	-	-	2	-	-	7	-	17	-	-	-	0	0	-	-	-	-
20	33	48	-	4	13	-	26	-	-	22	-	-	46	42	-	-	-	-	0	0	-	-	-
45	7	18	51	12	25	-	-	-	50	-	-	5	-	-	-	0	-	-	-	0	0	-	-
35	40	32	16	5	-	-	18	-	-	43	51	-	32	-	-	-	-	-	-	-	0	0	-
9	24	13	22	28	-	-	37	-	-	25	-	-	52	-	13	-	-	-	-	-	-	0	0
32	22	4	21	16	-	-	-	27	28	-	38	-	-	-	8	1	-	-	-	-	-	-	0

(c)

Code rate R= 3/4.

39	40	51	41	3	29	8	36	-	14	-	6	-	33	-	11	-	4	1	0	-	-	-	-
48	21	47	9	48	35	51	-	38	-	28	-	34	-	50	-	50	-	-	0	0	-	-	-
30	39	28	42	50	39	5	17	-	6	-	18	-	20	-	15	-	40	-	-	0	0	-	-
29	0	1	43	36	30	47	-	49	-	47	-	3	-	35	-	34	-	0	-	-	0	0	-
1	32	11	23	10	44	12	7	-	48	-	4	-	9	-	17	-	16	-	-	-	-	0	0
13	7	15	47	23	16	47	-	43	-	29	-	52	-	2	-	53	-	1	-	-	-	-	0

(d)

Code rate R= 5/6.

48	29	37	52	2	16	6	14	53	31	34	5	18	42	53	31	45	-	46	52	1	0	-	-
17	4	30	7	43	11	24	6	14	21	6	39	17	40	47	7	15	41	19	-	-	0	0	-
7	2	51	31	46	23	16	11	53	40	10	7	46	53	33	35	-	25	35	38	0	-	0	0
19	48	41	1	10	7	36	47	5	29	52	52	31	10	26	6	3	2	-	51	1	-	-	0

Table n105
Matrix prototypes of parity-check matrices for codeword block length *n*=1944 bits.
Subblock size is *Z* = 81 bits.

(a)
Code rate *R*= 1/2.

57	-	-	-	50	-	11	-	50	-	79	-	1	0	-	-	-	-	-	-	-	-	-	-
3	-	28	-	0	-	-	-	55	7	-	-	-	0	0	-	-	-	-	-	-	-	-	-
30	-	-	-	24	37	-	-	56	14	-	-	-	-	0	0	-	-	-	-	-	-	-	-
62	53	-	-	53	-	-	3	35	-	-	-	-	-	-	0	0	-	-	-	-	-	-	-
40	-	-	20	66	-	-	22	28	-	-	-	-	-	-	-	0	0	-	-	-	-	-	-
0	-	-	-	8	-	42	-	50	-	-	8	-	-	-	-	-	0	0	-	-	-	-	-
69	79	79	-	-	-	56	-	52	-	-	-	0	-	-	-	-	-	0	0	-	-	-	-
65	-	-	-	38	57	-	-	72	-	27	-	-	-	-	-	-	-	-	0	0	-	-	-
64	-	-	-	14	52	-	-	30	-	-	32	-	-	-	-	-	-	-	-	0	0	-	-
-	45	-	70	0	-	-	-	77	9	-	-	-	-	-	-	-	-	-	-	-	0	0	-
2	56	-	57	35	-	-	-	-	-	12	-	-	-	-	-	-	-	-	-	-	-	0	0
24	-	61	-	60	-	-	27	51	-	-	16	1	-	-	-	-	-	-	-	-	-	-	0

(b)

Code rate *R*= 2/3.

61	75	4	63	56	-	-	-	-	-	-	8	-	2	17	25	1	0	-	-	-	-	-	-
56	74	77	20	-	-	-	64	24	4	67	-	7	-	-	-	-	0	0	-	-	-	-	-
28	21	68	10	7	14	65	-	-	-	23	-	-	-	75	-	-	-	0	0	-	-	-	-
48	38	43	78	76	-	-	-	-	5	36	-	15	72	-	-	-	-	-	0	0	-	-	-
40	2	53	25	-	52	62	-	20	-	-	44	-	-	-	-	0	-	-	-	0	0	-	-
69	23	64	10	22	-	21	-	-	-	-	-	68	23	29	-	-	-	-	-	-	0	0	-
12	0	68	20	55	61	-	40	-	-	-	52	-	-	-	44	-	-	-	-	-	-	0	0
58	8	34	64	78	-	-	11	78	24	-	-	-	-	-	58	1	-	-	-	-	-	-	0

(c)

Code rate *R*= 3/4.

48	29	28	39	9	61	-	-	-	63	45	80	-	-	-	37	32	22	1	0	-	-	-	-
4	49	42	48	11	30	-	-	-	49	17	41	37	15	-	54	-	-	-	0	0	-	-	-
35	76	78	51	37	35	21	-	17	64	-	-	-	59	7	-	-	32	-	-	0	0	-	-
9	65	44	9	54	56	73	34	42	-	-	-	35	-	-	-	46	39	0	-	-	0	0	-
3	62	7	80	68	26	-	80	55	-	36	-	26	-	9	-	72	-	-	-	-	-	0	0
26	75	33	21	69	59	3	38	-	-	-	35	-	62	36	26	-	-	1	-	-	-	-	0

(d)

Code rate R= 5/6.

13	48	80	66	4	74	7	30	76	52	37	60	-	49	73	31	74	73	23	-	1	0	-	-
69	63	74	56	64	77	57	65	6	16	51	-	64	-	68	9	48	62	54	27	-	0	0	-
51	15	0	80	24	25	42	54	44	71	71	9	67	35	-	58	-	29	-	53	0	-	0	0
16	29	36	41	44	56	59	37	50	24	-	65	4	65	52	-	4	-	73	52	1	-	-	0

作者簡歷

陳美宇，畢業於逢甲大學電子工程學系。於民國九十七年取得碩士學位。家中成員有父母、姊姊、哥哥。

碩士研究方面，我主要研究 LDPC code Min-SumShuffled iterative decoding 之補償技術。