中正大學

通訊系統組專題

低密度同位檢查碼的

疊代解碼演算法的驗證

專題生: 顏浩恩

指導教授: 邱茂清

摘要

低密度同位檢查碼(LDPC)是一套可以在訊息傳送前編碼，然後經過通道加上了雜訊，在接收的時候可以透過特定的演算法可以有效解決錯誤並還原訊號的編碼。在本次專題，目標是理解、整理編碼的原理及過程，以及學習解碼的演算法，並透過程式模擬來驗證經過編碼的訊號，加入雜訊後，再解碼之後的錯誤率降低的結果。

目錄

第一章 簡介

低密度同位檢查碼(LDPC Codes)是屬於軟決策(soft dicision)的錯誤更正碼，軟決策是指解碼方式上，由接收端解調出來的是實數，代表著資料點的可靠度，這樣子的概率解碼通常在有資料受損的時候會有較好的結果。低密度同位檢查碼是區段碼(Block Codes)的一種，編碼器會接收一段位元資訊，轉換成一段傳送的位元，稱為碼字。低密度同位檢查碼的優點是存在置信傳播(belief propogation)的疊代解碼演算法。置信傳播是指用圖形模型來推斷訊息傳遞，這種演算法複雜度會隨著訊息長度線性增加。

在接下來的章節中，第二章從低密度同位檢查碼所屬的線性區段碼(Linear Block Code)開始認識，了解簡單的編碼方法。第三章是低密度同位檢查碼的表示方法。第四章是疊代演算法的概念簡介。第五章是機率領域的疊代演算法，第六章是對數域的疊代演算法。第七章是降低複雜度的演算法。第八章是模擬結果。

第二章 線性區段碼

2.1 線性碼編碼

線性碼就是將原始訊息加上額外用於偵錯的訊息組成碼字(codeword)。通常會用(n, k)表示，要傳送的訊息有k位元，額外加上的訊息有$n - k$位元，轉換後的碼字有 n 位元。每組n位元碼字可以看成n維的向量。而這種原始訊息有完整出現在轉換後碼字裡面的是屬於系統化碼(Systematic Code)，原始訊息稱為訊息位元(message bits)，額外訊息稱為同位位元(parity bits)或是多於位元(redundant bits)。

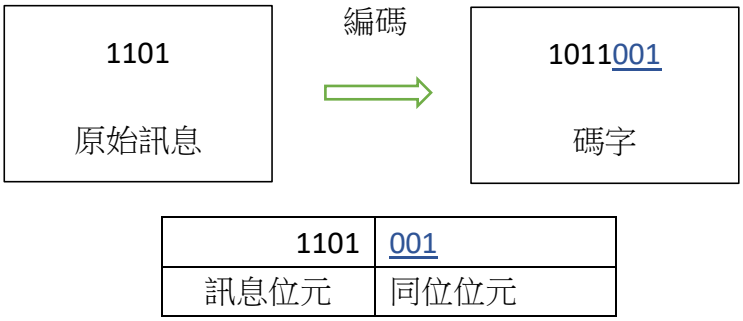


圖. (7,4)線性碼編碼範例

2.2 編碼率

線性碼的編碼率(Code's Rate)是編碼前位元數除以編碼後位元數，所以(n, k)線性碼的編碼率就是k/n。上圖的例子$(7,4)$線性碼的編碼率就是$4/7$。

2.3 生成矩陣(Generator Matrix)

一個(n, k)的線性碼是k維的向量空間，可用一組線性獨立的碼字

$\vec{g}_1, \vec{g}_2, \ldots, \vec{g}_k$來表示。這k個碼字組成的$k \times n$的矩陣G，稱為生成矩陣。

$$G = \begin{bmatrix} \vec{g}_1 \\ \vec{g}_2 \\ \vdots \\ \vec{g}_k \end{bmatrix} = \begin{bmatrix} \vec{g}_{11} & \vec{g}_{12} & \cdots & \vec{g}_{1n} \\ \vec{g}_{21} & \vec{g}_{22} & \cdots & \vec{g}_{2n} \\ & \vdots & & \\ \vec{g}_{k1} & \vec{g}_{k2} & \cdots & \vec{g}_{kn} \end{bmatrix}$$

圖. 生成矩陣

編碼時就是由原始訊息$\vec{m} = (m_1, m_2, \ldots, m_k)$乘上生成矩陣可以得到碼字$\vec{m}G = \vec{x} = (x_1, x_2, \ldots, x_n)$。

2.4 對偶碼(Duel Code)與同位檢查(Parity Check)方程式

令 C 是一(n, k)線性碼，對 C 的同位檢查方程式為$p_1x_1 + p_2x_2 + p_3x_3 + \cdots + p_nx_n = 0$，此方程式對所有屬於 C 的線性碼都成立。而滿足同位檢查方程式的向量$(p_1, p_2, p_3, \ldots, p_n)$的集合會形成 $n - k$ 維的向量子空間，以 $C^{\perp}$(C perp)表示，就是 C 的對偶碼。

2.5 同位檢查矩陣(Parity-Check Matrix)

令 C 是一(n, k)線性碼， $C^{\perp}$ 是 C 的對偶碼，則 $C^{\perp}$ 的一組基底向量 $\overrightarrow{h_1}, \overrightarrow{h_2}, \overrightarrow{h_3}, \ldots, \overrightarrow{h_{n-k}}$ 組成的矩陣 H 就是同位檢查矩陣。此矩陣滿足 $H\vec{x}^T = 0$，$\vec{x} \in C$。矩陣 H 為 C 的同位檢查矩陣，也為 $C^{\perp}$ 的生成矩陣。

$$H = \begin{bmatrix} \overrightarrow{h_1} \\ \overrightarrow{h_2} \\ \overrightarrow{h_3} \\ \vdots \\ \overrightarrow{h_{n-k}} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & \cdots & h_{1,n} \\ h_{21} & h_{22} & h_{23} & \cdots & h_{2,n} \\ h_{31} & h_{32} & h_{33} & \cdots & h_{3,n} \\ & & \vdots & & \\ h_{n-k,1} & h_{n-k,2} & h_{n-k,3} & \cdots & h_{n-k,n} \end{bmatrix}$$

圖. (n, k)線性碼的同位檢查矩陣

2.6 (7, 4)漢明碼(Hamming Code)

(7,4)漢明碼編碼方式是先將原始訊號的四個二進位數字依序排入#1, #2, #3, #4，接著在#5, #6, #7 也填入二進位數字使數字滿足三個圓內數字的總和為偶數。

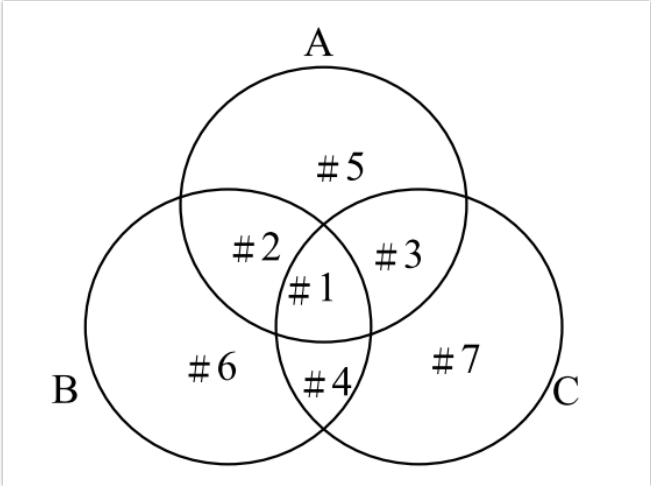


圖. 漢明碼編碼用圖形

以上規則如果將x1, x2, x3, x4, x5, x6, x7依序代表填入的數字，可以寫成三個一位元的二進位加法的同位檢查方程式:

$$\begin{cases} x_1 + x_2 + x_3 + x_5 = 0 \\ x_1 + x_2 + x_4 + x_6 = 0 \\ x_1 + x_3 + x_4 + x_7 = 0 \end{cases}$$

其同位檢查矩陣就可以寫為:

$$\mathrm{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

2.7 漢明重量與漢明距離(Hamming weight, Hamming distance)

一個碼字的漢明重量就是碼字中非零分量的個數。例如 $\vec{\mathrm{x}} = (1,0,0,1,1,0,1)$

的漢明重量就是 4。而漢明距離就是兩個碼字當中相異分量的個數。例如 $\vec{\mathrm{x}} = (1,0,0,1,1,0,1)$，$\vec{\mathrm{y}} = (1,1,1,1,1,1,1)$ 的漢明距離就是 3。

2.8 徵狀(Syndrome)

若傳送的碼字是$\vec{\mathrm{x}} = (x_1, x_2, x_3, \dots, x_n)$，接收的碼字是$\vec{\mathrm{y}} = (y_1, y_2, y_3, \dots, y_n)$則它們的差 $\vec{\mathrm{y}} - \vec{\mathrm{x}} = \vec{\mathrm{e}} = (e_1, e_2, e_3, \dots, e_n)$ 稱為誤差向量(error pattern)。我們可以利用同位檢查矩陣用在解碼上。若 H 是 $\vec{\mathrm{x}}$ 的同位檢查矩陣，$\mathrm{H}\vec{\mathrm{x}}^T = \vec{0}$，而乘上接收向量 $\vec{\mathrm{s}} = \mathrm{H}\vec{\mathrm{y}}^T$，$\vec{\mathrm{s}}$就稱為徵狀。

當錯誤發生時誤差向量 $\vec{\mathrm{e}} \neq \vec{0}$，而也很有可能同為矩陣乘上接收的向量 $\mathrm{H}\vec{\mathrm{y}}^T \neq 0$，而徵狀 $\vec{\mathrm{s}} = \mathrm{H}\vec{\mathrm{y}}^T = \mathrm{H}(\vec{\mathrm{x}} + \vec{\mathrm{e}})^T = \mathrm{H}\vec{\mathrm{x}}^T + \mathrm{H}\vec{\mathrm{e}}^T = \mathrm{H}\vec{\mathrm{e}}^T$。

簡單的徵狀解碼是利用徵狀來查表來找出可能的誤差向量，再將接收到的向量減去估計的誤差向量來做出修正。

第三章 LDPC 介紹

3.1 簡介

低密度同位檢查碼(Low-Density Parity-Check Codes)，簡稱 LDPC 碼，是一種線性區段碼。低密度是指在它的同位檢查矩陣中，非零的元素比例很低，這樣的矩陣是稀疏矩陣(sparse matrix)。同位檢查矩陣的每一行是一個同位檢查方程式，可以用於檢查出錯誤。LDPC 還是錯誤更正碼，可以在接收端解碼時運用特殊演算法來更正錯誤並盡可能地還原。其優點是可以在大量傳輸時，提供接近通道容量的效能。

3.2 規則(Regular)與不規則(Irregular)

按照同位檢查矩陣的行與列的漢明重量是否為定數來分別規則 LDPC 與不規則 LDPC。規則 LDPC 的同位矩陣條件有三條:

(1)每行皆有ω_c個 1。

(2)每列皆有ω_r個 1， 且 $\frac{\omega_c}{\omega_r}=\frac{m}{n}$。 (m 為總列數，n 為總行數)

(3) $\omega_c \ll m$，$\omega_r \ll n$。

其編碼率 $\mathrm{R}=\frac{k}{n}=\frac{n-m}{n}=1-\frac{m}{n}=1-\frac{\omega_c}{\omega_r}$。

而不規則 LDPC 的同位矩陣的任兩行或任兩列的 1 的個數不等，且行及列的 1 的個數遠小於列及行的個數，即滿足條件。

3.3 LDPC 的圖形表示(Tanner Graph)

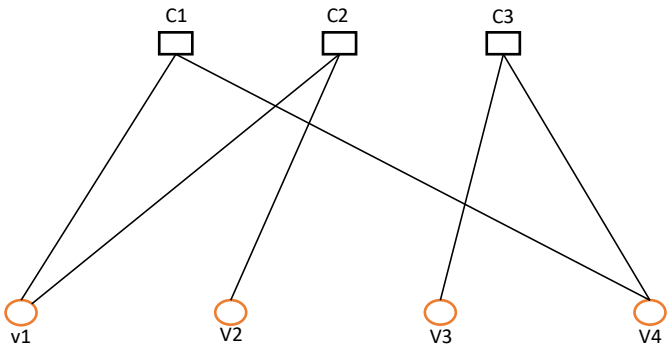
Tanner 圖是由節點(node)跟邊(edge)組成，節點有兩種，一個是變數節點

(variable node)又稱為位元節點(bit node)，另一種是檢查節點(check node)。第 n 個變數節點代表著同位檢查矩陣的第 n 行，也是碼字的第 n 個位元。第 m 個檢查節點代表同位檢查矩陣第 m 列，也是第 m 個同位檢查方程式。而邊則是連接檢查節點與變數節點。當同位檢查矩陣的第 j 行第 k 列為 1，在 tanner 圖上第 j 個檢查節點c_j就與第 k 個變數節點v_k連成邊。

範例 (連接方式，非低密度)：

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

同位檢查矩陣



Tanner 圖

3.4 度分佈多項式 (Degree Distribution Polynomials)

從 Tanner 圖來看，度數(degree)用來表示一個節點有幾個邊連結。度分佈多項式通常用來表示不規則 LDPC 碼。我們可以用序列$\{\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{d_v}\}$與$\{\rho_1, \rho_2, \rho_3, \dots, \rho_{d_c}\}$來表示邊的度分佈。 λ_i 表示度數為 i 的變數節點占總邊數的比例，ρ_j 表示度數為 j 的檢查節點占總邊數的比例。d_v 是變數節點的最大度數，d_c 是檢查節點的最大度數。

第四章 疊代解碼演算法簡介

(Iterative Decoding Algorithm)

4.1 傳送訊號的種類

從傳送端送出向量 $\vec{\mathrm{x}} = (x_1, x_2, x_3, \dots, x_n)$，在接收端接收到向量 $\vec{\mathrm{y}} = (y_1, y_2, y_3, \dots, y_n)$，疊代解碼就是要在接收端去計算 $\vec{\mathrm{x}}$ 中每個元素 x_j 為 1 的機率。在疊代的過程，疊代是指訊息在變數節點與檢查節點之間來回傳遞，而傳遞的訊息有三個種類：

- 後驗機率 APP (a posteriori probability)：

$$\Pr(\overrightarrow{x_j} = 1|\overrightarrow{y_j})$$

- 後驗機率比 (APP ratio)，又叫概似比 LR (likelihood ratio)：

$$\mathrm{l}(\overrightarrow{x_j}) \triangleq \frac{\Pr(\overrightarrow{x_j} = 0|\overrightarrow{y_j})}{\Pr(\overrightarrow{x_j} = 1|\overrightarrow{y_j})}$$

- 後驗機率比對數 (log-APP ratio)，又叫概似比對數 LLR(log-likelihood ratio)：

$$\mathrm{LLR}(\overrightarrow{x_j}) \triangleq \log\left(\frac{\Pr(\overrightarrow{x_j} = 0|\overrightarrow{y_j})}{\Pr(\overrightarrow{x_j} = 1|\overrightarrow{y_j})}\right)$$

這裡的 log 是以自然對數 e 為底的。

4.2 訊息除傳遞演算法 MPA (Message Passing Algorithm)

在計算$\Pr(\overrightarrow{x_j} = 1|\overrightarrow{y_j})$、$\mathrm{l}(\overrightarrow{x_j})$、$\mathrm{LLR}(\overrightarrow{x_j})$時的訊息傳遞演算法以 Tanner 圖為基礎，將兩種節點，變數節點以及檢查節點當作兩種處理器，每個節點代表一個處理器，來處理兩種訊息。而圖上連接節點的邊就作為訊息傳遞的路徑。在畫 Tanner 圖時習慣將檢查節點放在變數節點的上方。

4.3 疊代：變數節點到檢查節點 (V-node to C-node)

變數節點將處理完的訊息傳遞到檢查節點的過程，從 Tanner 圖上的畫法是向上傳遞，由第 j 個變數節點傳到第 k 個檢查節點的訊息可以表示成$m_{\uparrow jk}$，每一個邊所相連的檢查節點都會接收到對應的由變數節點所傳遞的訊息。而從第 j 個變數節點所傳出的訊息是條件機率$\Pr(\overrightarrow{x_j} = \mathrm{b} \mid \text{輸入的訊息})$，$\mathrm{b} \in \{0,1\}$，或是其機率比(LR)或是對數機率比(LLR)。

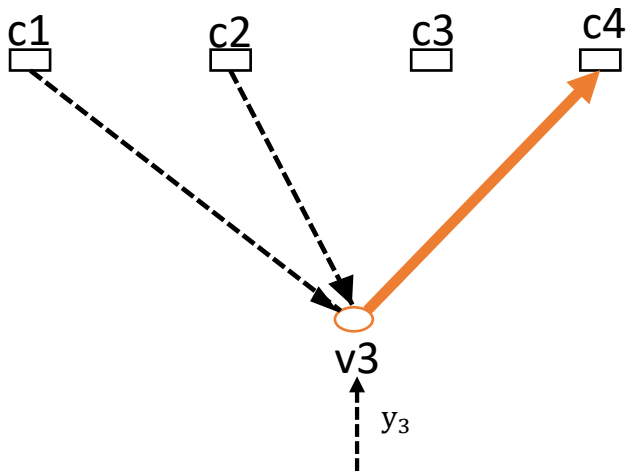


圖. 由 v3 傳遞訊息到 c4

上圖是對應同位檢查矩陣的第三行$\begin{bmatrix}1\\1\\0\\1\end{bmatrix}$，其步驟是傳遞$m_{\uparrow 34}$，而$m_{\uparrow 34}$是由 v3 處理 c1、c2 (c4 除外)及通道傳遞的 y3 所計算出來的。

4.4 疊代：檢查節點到變數節點 (C-node to V-node)

檢查節點將處理完的訊息傳遞到變數節點的過程，從 Tanner 圖上的畫法是向下傳遞，由第 k 個檢查節點傳到第 j 個變數節點的訊息可以表示成$m_{\downarrow kj}$，每一個邊所相連的變數節點都會接收到對應的由檢查節點所傳遞的訊息。而從第 k 個變數節點所傳出的訊息是條件機率$\Pr(\text{滿足同位檢查方程式}\mathrm{C}_k \mid \overrightarrow{x_j} =$

b、輸入的訊息)，b ∈ {0,1}，或是其機率比(LR)或是對數機率比(LLR)。

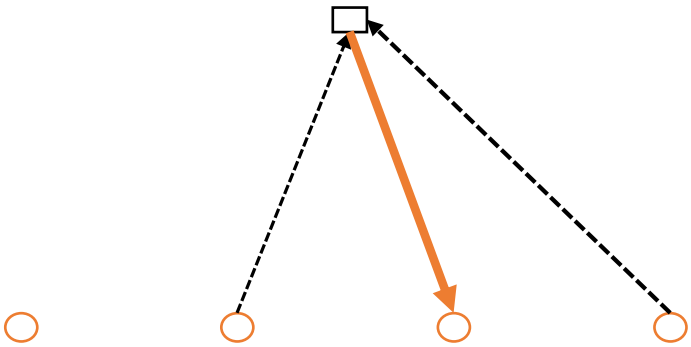


圖. 由 c2 傳遞訊息到 v3

上圖是對應同位檢查矩陣的第二列[0　1　1　1]，其步驟是傳遞$m_{\downarrow 23}$，而$m_{\downarrow 23}$是由 c2 處理 v2、v4 (v3 除外)所計算出來的。

4.5 疊代決策

我們會設定一個最大疊代次數，或是一個停止疊代條件像是解碼的碼字 $\vec{x}$ 滿足$H\vec{x} = 0$，就會停止疊代，然後對解碼的值做出最後決策。

第五章 機率領域的和積演算法

(Probability-Domain Sum-Product Algorithm)

5.1 符號定義

- C_j ：表示所有連接到第 j 個變數節點v_j的檢查節點的集合。
- V_i ：表示所有連接到第 i 個變數節點c_i的變數節點的集合。
- $C_j \backslash i$ ：表示C_j，且排除掉第 i 個檢查節點c_i的集合。
- $V_i \backslash j$ ：表示V_i，且排除掉第 j 個檢查節點v_j的集合。
- P_j ：$\Pr(x_j = 1|y_j)$，在接收到第 j 個位元是y_j時，傳送的第 j 個位元是 1 的機率。
- $1 - P_j$ ：$\Pr(x_j = 0|y_j)$，在接收到第 j 個位元是y_j時，傳送的第 j 個位元是 0 的機率。

5.2 計算 q 值

將 q 值設定成$q_{ij}(b) = \{x_j = b \mid y_j$，*滿足含有*$x_j$*的檢查方程式*$c_{i'}$，$i' \neq i\}$，$b \in \{0,1\}$，$y_j$是通道訊息，滿足含有$x_j$的檢查方程式$c_{i'}$是來自於檢查節點的訊息。q 值可以表示為：

$$q_{ij}(0) = K_{ij}(1 - P_j) \prod_{i' \in C_i \backslash i} r_{i'j}(0)$$

$$q_{ij}(1) = K_{ij} P_j \prod_{i' \in C_j \backslash i} r_{i'j}(1)$$

K_{ij}為常數，在計算後設定使$q_{ij}(0) + q_{ij}(1) = 1$。

5.3 設定初始值

初始值的設定是設定 q 值，$q_{ij}(b) = \Pr(x_j = b \mid y_j)$，會根據不同的通道有不同的設定公式。在這裡介紹的是根據二元輸入加性高斯白雜訊通道(binary input additive white Gaussian noise channel, BI-AWGNC)來設定。

傳送的位元為x_j，接收的位元為y_j，令$t_j = 1 - 2x_j$，將$x_j = 0(1)$轉換成$t_j = 1(-1)$，而$y_j = t_j + n_j$，雜訊n_j是呈$\eta(0, \sigma^2)$常態分佈，初始值為：

$$\Pr(t_j = t \mid y_j) = \frac{1}{1 + exp\left(-2y_j t/\sigma^2\right)} \quad , \quad t \in \{1, -1\}$$

證明：

$$\Pr(t_j = t \mid y_j) = \frac{\Pr(t_j = t)}{\Pr(y_j)}$$

$$= \frac{\Pr(t_j = t)}{\Pr(t_j = t \mid y_j) + \Pr(t_j = -t \mid y_j)}$$

$$= \frac{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_j - t)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_j - t)^2}{2\sigma^2}\right) + \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_j + t)^2}{2\sigma^2}\right)}$$

$$= \frac{1}{1 + \exp\left(-\frac{(y_j + t)^2}{2\sigma^2}\right) / \exp\left(-\frac{(y_j - t)^2}{2\sigma^2}\right)}$$

$$= \frac{1}{1 + exp\left(-2y_j t/\sigma^2\right)}$$

5.4 計算 r 值

r 值是指傳送由檢查節點c_i到變數節點v_j的訊息$m_{\downarrow ij}$，包含$r_{ij}(b)$，$b \in \{0,1\}$是由c_i的外來訊息(連到c_i的變數節點，不包含v_j)計算而得的。

- $r_{ij}(0)$：傳送第 j 個位元$x_j = 0$滿足檢查方程式c_i的機率。
- $r_{ij}(1)$：傳送第 j 個位元$x_j = 1$滿足檢查方程式c_i的機率。

導出$r_{ij}(b)$，需要以下引理：

引理 1：設數列$\{x_j\}_{j=1}^{M}$有 M 個獨立的二進位數字，且$P_r(x_j = 1) = P_j$，則此數列有偶數個 1 的機率為：

$$\frac{1}{2} + \frac{1}{2}\prod_{j=1}^{M}(1 - 2P_j)$$

證明：

令 F(t) $= \prod_{j=1}^{M}[(1 - P_j) + P_j t]$，F(t)展開來為 t 的多項式，多項式中$t^j$的係數為含 j 個 1 的機率，所以偶數個 1 的機率就是多項式 t 的偶次項係數的和。當 F(t)與 F(-t)相減，就會將奇次項的係數消去，所以數列中偶數個 1 的機率就是：

$$\frac{F(1) - F(-1)}{2} = \frac{\prod_{j=1}^{M}[(1 - P_j) + P_j] + \prod_{j=1}^{M}[(1 - P_j) - P_j]}{2}$$

$$= \frac{1}{2} + \frac{1}{2}\prod_{j=1}^{M}(1 - 2P_j)$$

現在將引理 1 中的P_j替換成$q_{ij'}(1)$，訊息$q_{ij'}(1)$即是每個外來位元 $x_{j'} = 1$ 的機率。當外來位元有偶數個 1，輸出為 0；若外來位元有奇數個 1，輸出為 1，以滿足同位檢查方程式。所以r_{ij}可以表示為：

$$r_{ij}(0) = \frac{1}{2} + \frac{1}{2}\prod_{j' \in V_i \backslash j}\left(1 - 2q_{ij'}(1)\right)$$

而 $$r_{ij}(1) = 1 - r_{ij}(0)$$

5.5 計算 Q 值

Q 值是進行完疊代用來對解碼的值做出決策的值，其數量跟碼字的數量相

同，每次疊代完更新一次。計算如下：

$$Q_j(0) = K_j(1 - P_j)\prod_{i\in C_j} r_{ij}(0)$$

$$Q_j(1) = K_j P_j \prod_{i\in C_j} r_{ij}(1)$$

K_j為常數，在計算後設定使$Q_j(0) + Q_j(1) = 1$。

5.6 疊代步驟整理

Step1. 設定初始值$q_{ij}(0)$、$q_{ij}(1)$

對所有同位檢查矩陣 H 的元素h_{ij}檢查，h_{ij}為 1 時，設定$q_{ij}(b) = \Pr(x_j = b \mid y_j)$。

Step2. 更新$r_{ij}(0)$、$r_{ij}(1)$

Step3. 更新$q_{ij}(0)$、$q_{ij}(1)$

Step4. 更新$Q_j(0)$、$Q_j(1)$

Step5. 決策

$$\hat{x}_j = \begin{cases} 1 & if \;\; Q_j(1) > Q_j(0) \\ 0 & if \;\; Q_j(0) > Q_j(1) \end{cases}$$

Step6. 疊代持續判斷

當疊代超過最大次數或是解碼滿足$\mathrm{H}\vec{\mathrm{x}} = 0$，停止疊代。

第六章 對數領域和積演算法

(Log-Domain Sum-Product Algorithm)

6.1 簡介

在機率領域和積演算法中，因為計算過程大部分是連續乘積，容易造成數字上的不穩定，如果將演算法轉換成對數領域，就可以轉換成連加運算，因此對數領域和積演算法相對較受歡迎。其疊代步驟與機率領域的相同。

6.2 符號定義

- $\mathrm{LLR}(x_j) = \log\left(\frac{\Pr(x_j=0|y_j)}{\Pr(x_j=1|y_j)}\right)$
- $\mathrm{LLR}(r_{ij}) = \log\left(\frac{r_{ij}(0)}{r_{ij}(1)}\right)$
- $\mathrm{LLR}(q_{ij}) = \log\left(\frac{q_{ij}(0)}{q_{ij}(1)}\right)$
- $\mathrm{LLR}(Q_j) = \log\left(\frac{Q_j(0)}{Q_j(1)}\right)$

6.3 初始值設定

將機率領域轉換過來變成

$$\mathrm{LLR}(q_{ij}) = \mathrm{LLR}(x_j) = \frac{2y_i}{\sigma^2}$$

證明：

傳送的位元為x_j，接收的位元為y_j，令$t_j = 1 - 2x_j$， $y_j = t_j + n_j$。又

$$\Pr(t_j = t \mid y_j) = \frac{1}{1 + exp(-2y_j t/\sigma^2)}$$

$$\mathrm{LLR}(q_{ij}) = \mathrm{LLR}(x_j) = log\left(\frac{\Pr(x_j = 0|y_j)}{\Pr(x_j = 1|y_j)}\right)$$

$$= log\left(\frac{\Pr(t_j = 1 \mid y_j)}{\Pr(t_j = -1 \mid y_j)}\right)$$

$$= log\left(\frac{1 + exp(2y_j/\sigma^2)}{1 + exp(-2y_j/\sigma^2)}\right)$$

$$= log\left(exp(2y_j/\sigma^2)\right) = 2y_j/\sigma^2$$

6.4 計算$\mathbf{LLR}(q_{ij})$

機率領域的 q 值：

$$q_{ij}(0) = K_{ij}(1 - P_j) \prod_{i' \in C_j \backslash i} r_{i'j}(0)$$

$$q_{ij}(1) = K_{ij} P_j \prod_{i' \in C_j \backslash i} r_{i'j}(1)$$

$$\mathrm{LLR}(q_{ij}) = \log\left(\frac{q_{ij}(0)}{q_{ij}(1)}\right) = \log\left(\frac{(1 - P_j)\prod_{i' \in C_j \backslash i} r_{i'j}(0)}{P_j \prod_{i' \in C_j \backslash i} r_{i'j}(1)}\right)$$

$$= \log\left(\frac{(1 - P_j)}{P_j}\right) + \log\left(\frac{\prod_{i' \in C_j \backslash i} r_{i'j}(0)}{\prod_{i' \in C_j \backslash i} r_{i'j}(1)}\right)$$

$$= \log\left(\frac{\Pr(x_j = 0|y_j)}{\Pr(x_j = 1|y_j)}\right) + \sum_{i' \in C_j \backslash i} \log\left(\frac{r_{i'j}(0)}{r_{i'j}(1)}\right)$$

$$= \mathrm{LLR}(x_j) + \sum_{i' \in C_j \backslash i} \mathrm{LLR}(r_{i'j})$$

6.5 計算LLR(r_{ij})

在導出的過程需要以下引理：

引理 2：若$p_0 + p_1 = 1$，則$\tanh\left(\frac{1}{2}\log\frac{p_0}{p_1}\right) = 1 - 2p_1$。

證明：

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

$$\therefore \tanh\left(\frac{1}{2}\log\frac{p_0}{p_1}\right) = \frac{e^{2\log\frac{p_0}{p_1}} - 1}{e^{2\log\frac{p_0}{p_1}} + 1}$$

$$= \frac{\frac{p_0}{p_1} - 1}{\frac{p_0}{p_1} + 1} = \frac{p_0 - p_1}{p_0 + p_1} = p_0 - p_1$$

$$= 1 - 2p_1$$

引理 3：若 $\phi(x) = -\log\left(\tanh\frac{x}{2}\right)$ ，且$x > 0$，則$\phi(x) = \phi^{-1}(x)$。

證明：

$$\phi(x) = -\log\left(\tanh\frac{x}{2}\right) = -\log\frac{e^{\frac{x}{2}} - e^{-\frac{x}{2}}}{e^{\frac{x}{2}} + e^{-\frac{x}{2}}} = -\log\frac{e^x - 1}{e^x + 1} = \log\frac{e^x + 1}{e^x - 1}$$

$$\Rightarrow \phi\left(\phi(x)\right) = \log\frac{e^{\phi(x)} + 1}{e^{\phi(x)} - 1} = \log\left(\frac{\frac{e^x + 1}{e^x - 1} + 1}{\frac{e^x + 1}{e^x - 1} - 1}\right)$$

$$= \log\left(\frac{(e^x + 1) + (e^x - 1)}{(e^x + 1) - (e^x - 1)}\right)$$

$$= \log e^x = x$$

$$\Rightarrow \phi(x) = \phi^{-1}(x)$$

接下來是算$\mathrm{LLR}(r_{ij})$的過程：

機率領域的 r 值：

$$r_{ij}(0) = \frac{1}{2} + \frac{1}{2}\prod_{j' \in V_i \backslash j}\left(1 - 2q_{ij'}(1)\right)$$

$$r_{ij}(1) = 1 - r_{ij}(0)$$

$$\Rightarrow 1 - r_{ij}(1) = \frac{1}{2} + \frac{1}{2}\prod_{j' \in V_i \backslash j}\left(1 - 2q_{ij'}(1)\right)$$

$$\Rightarrow \frac{1}{2} - r_{ij}(1) = \frac{1}{2}\prod_{j' \in V_i \backslash j}\left(1 - 2q_{ij'}(1)\right)$$

$$\Rightarrow 1 - 2r_{ij}(1) = \prod_{j' \in V_i \backslash j}\left(1 - 2q_{ij'}(1)\right)$$

利用引理 2 $\quad \Rightarrow \tanh\left(\frac{1}{2}\log\frac{r_{ij}(0)}{r_{ij}(1)}\right) = \prod_{j' \in V_i \backslash j}\left(\frac{1}{2}\log\frac{\mathrm{q}_{ij'}(0)}{q_{ij'}(1)}\right)$

$$\Rightarrow \tanh\left(\frac{1}{2}\mathrm{LLR}(r_{ij})\right) = \prod_{j' \in V_i \backslash j}\tanh\left(\frac{1}{2}\log\frac{\mathrm{q}_{ij'}(0)}{q_{ij'}(1)}\right)$$

$$\Rightarrow \mathbf{LLR}(\boldsymbol{r_{ij}}) = 2\tanh^{-1}\Big(\prod_{j' \in V_i \backslash j}\tanh\left(\frac{1}{2}\mathrm{LLR}(\mathrm{q}_{ij'})\right)\Big)$$，令 $\mathrm{LLR}(\mathrm{q}_{ij'}) = \alpha_{ij'}\beta_{ij'}$

$$= 2\tanh^{-1}\Big(\prod_{j' \in V_i Jay}\tanh\left(\frac{1}{2}\alpha_{ij'}\beta_{ij'}\right)\Big)$$

(其中$\alpha_{ij'} = \mathrm{sign}\left(\mathrm{LLR}(\mathrm{q}_{ij'})\right); \beta_{ij'} = |\,\mathrm{LLR}(\mathrm{q}_{ij'})|$)

$$= \prod_{j' \in V_i \backslash j}\alpha_{ij'}\, 2\tanh^{-1}\Big(\prod_{j' \in V_i \backslash j}\tanh\left(\frac{1}{2}\beta_{ij'}\right)\Big)$$

$$= \prod_{j' \in V_i \backslash j} \alpha_{ij'} \, 2\tanh^{-1}\log^{-1}\log\left(\prod_{j' \in V_i \backslash j} \tanh\left(\frac{1}{2}\beta_{ij'}\right)\right)$$

$$= \prod_{j' \in V_i \backslash j} \alpha_{ij'} \, 2\tanh^{-1}\log^{-1} \sum \log\left(\tanh\left(\frac{1}{2}\beta_{ij'}\right)\right)$$

$$(\text{令 } \phi(x) = -\log\left(\tanh\frac{x}{2}\right))$$

$$= \prod_{j' \in V_i \backslash j} \alpha_{ij'} \, \phi^{-1}\left(\sum_{j' \in V_i \backslash j} \phi(\beta_{ij'})\right)$$

利用引理 3 ⇒ $\mathbf{LLR}(\boldsymbol{r_{ij}}) = \prod_{j' \in V_i \backslash j} \alpha_{ij'} \, \phi\left(\sum_{j' \in V_i \backslash j} \phi(\beta_{ij'})\right)$

$$= \prod_{j' \in V_i \backslash j} \text{sign}\left(\text{LLR}(q_{ij'})\right) \phi\left(\sum_{j' \in V_i \backslash j} \phi(|\,\text{LLR}(q_{ij'})|)\right)$$

6.6 計算 Q 值

以機率領域的 Q 值，簡化過程如 q 值，結果如下：

$$\text{LLR}(Q_j) = \text{LLR}(x_j) + \sum_{i \in C_j} \text{LLR}(r_{ij})$$

第七章 降低複雜度解碼演算法

7.1 簡介

降低複雜度的解碼演算法是用不同的觀點來詮釋，將原本的對數領域和積演算法做出修改，來降低複雜度，但是在效能上有一些損失。修改的部分是對數領域和積演算法裡的$\mathrm{LLR}(r_{ij})$值的計算過程，在推導前會先有三個引理證明。

7.2 符號定義

- $\mathrm{p}_{10} = \Pr(\mathrm{a}_1 = 0)$，$\mathrm{p}_{11} = \Pr(\mathrm{a}_1 = 1)$，$\mathrm{p}_{20} = \Pr(\mathrm{a}_2 = 0)$，$\mathrm{p}_{10} = \Pr(\mathrm{a}_2 = 1)$
- $\mathrm{L_i} = \mathrm{LLR}(a_i) = \log\frac{\Pr(a_i = 0)}{\Pr(a_i = 1)} = \log\left(\frac{\mathrm{p_{i0}}}{\mathrm{p_{i1}}}\right)$表示變數$a_i$之概似比對數(LLR)
- $\mathrm{a}_1 \oplus \mathrm{a}_2$表示$\mathrm{a}_1$與$\mathrm{a}_2$的二進位的和，即$\mathrm{a}_1 + \mathrm{a}_2$的和除以 2 之餘數。
- $\mathrm{L}_1 \boxplus \mathrm{L}_2 = \mathrm{LLR}(\mathrm{a}_1 \oplus \mathrm{a}_2) = \log\frac{\Pr(\mathrm{a}_1 \oplus \mathrm{a}_2 = 0)}{\Pr(\mathrm{a}_1 \oplus \mathrm{a}_2 = 1)}$

7.3 引理及證明

引理 4：設a_1與a_2是兩二進位變數，且L_1與L_2分別是a_1與a_2的 LLR，若$\mathrm{L}_1 \boxplus \mathrm{L}_2$是$\mathrm{a}_1 \oplus \mathrm{a}_2$的 LLR 則

$$\mathrm{L}_1 \boxplus \mathrm{L}_2 = \log\left(\frac{1 + e^{L_1 + L_2}}{e^{L_1} + e^{L_2}}\right)$$

證明：

$$\mathrm{L}_1 \boxplus \mathrm{L}_2 = \mathrm{LLR}(\mathrm{a}_1 \oplus \mathrm{a}_2)$$

$$= \log\frac{\Pr(\mathrm{a}_1 \oplus \mathrm{a}_2 = 0)}{\Pr(\mathrm{a}_1 \oplus \mathrm{a}_2 = 1)}$$

$\mathrm{a}_1 \oplus \mathrm{a}_2 = 0 \Rightarrow (\mathrm{a}_1 = 0$ 且$\mathrm{a}_2 = 0)$或$(\mathrm{a}_1 = 1$ 且$\mathrm{a}_2 = 1)$

$\mathrm{a}_1 \oplus \mathrm{a}_2 = 1 \Rightarrow (\mathrm{a}_1 = 0$ 且$\mathrm{a}_2 = 1)$或$(\mathrm{a}_1 = 1$ 且$\mathrm{a}_2 = 0)$

$$= \log \frac{p_{10}p_{20} + p_{11}p_{21}}{p_{10}p_{21} + p_{11}p_{20}} = \log \left(\frac{\frac{p_{10}}{p_{11}} \frac{p_{20}}{p_{21}} + 1}{\frac{p_{10}}{p_{11}} + \frac{p_{20}}{p_{21}}} \right)$$

$$= \log \left(\frac{e^{L_1} e^{L_2} + 1}{e^{L_1} + e^{L_2}} \right) = \log \left(\frac{1 + e^{L_1 + L_2}}{e^{L_1} + e^{L_2}} \right)$$

引理 5：對任意兩實數x, y，定義$\max^*(x, y) = \log(e^x + e^y)$，則

$$\max^*(x, y) = \max(x, y) + \log(1 + e^{-|x-y|})$$

證明：

若$x \geq y$，$\max^*(x, y) = \log(e^x + e^y) = \log e^x (1 + e^{-(x-y)})$

$$= \log e^x + \log(1 + e^{-(x-y)})$$

$$= x + \log(1 + e^{-|x-y|})$$

$$= \max(x, y) + \log(1 + e^{-|x-y|})$$

若$x \leq y$，同理可證。

引理 6：x, y為實數，則$\max(0, x + y) - \max(x, y) = \text{sign}(x)\text{sign}(y)\min(|x|, |y|)$

證明：

若 $x + y > 0$ 且 $x > y \Rightarrow x > y > -x \Rightarrow |x| > |y|$且 $x > 0$

$\Rightarrow$ 左式 $= x + y - x = y =$ $\text{sign}(x)\text{sign}(y)\min(|x|, |y|)$

若 $x + y > 0$ 且 $y > x$，同理可證。

若 $x + y < 0$ 且 $x > y \Rightarrow -y > x > y \Rightarrow |x| < |y|$且 $y < 0$

$\Rightarrow$ 左式 $= 0 - x = -x =$ sign(x)sign(y)min(|x|, |y|)

若 $x + y < 0$ 且 $y > x$，同理可證。

7.4 新 r 值推導

因為$\mathrm{LLR}(r_{ij})$的計算式相當於求二進位的隨機變數的和的 LLR，且在隨機變數為獨立的假設下，根據引理 4 可以得到

$\mathrm{LLR}(r_{ij}) = \mathrm{LLR}(q_{ij_1'}) \boxplus \mathrm{LLR}(q_{ij_2'}) \boxplus \mathrm{LLR}(q_{ij_3'}) \ldots \boxplus \mathrm{LLR}(q_{ij_\omega'})$，其中$j_t' \epsilon V_i \backslash j$

而其計算過程式根據引理 4、引理 5：

$$\mathrm{L}_1 \boxplus \mathrm{L}_2 = \log\left(\frac{1 + e^{L_1+L_2}}{e^{L_1} + e^{L_2}}\right) = \log\left(\frac{e^0 + e^{L_1+L_2}}{e^{L_1} + e^{L_2}}\right)$$

$$= \log(e^0 + e^{L_1+L_2}) - \log(e^{L_1} + e^{L_2})$$

$$= \max^*(0, L_1 + L_2) - \max^*(L_1, L_2)$$

$$= \left[\max(0, L_1 + L_2) + \log(1 + e^{-|L_1+L_2|})\right] - \left[\max(L_1, L_2) + \log(1 + e^{-|L_1-L_2|})\right]$$

$$= [\max(0, L_1 + L_2) - \max(L_1, L_2)] + \left[\log(1 + e^{-|L_1+L_2|}) - \log(1 + e^{-|L_1-L_2|})\right]$$

令 $\mathrm{s(x,y)} = \log(1 + e^{-|L_1+L_2|}) - \log(1 + e^{-|L_1-L_2|})$

$$= [\max(0, L_1 + L_2) - \max(L_1, L_2)] + \mathrm{s(x,y)}$$

這裡再證明$|\mathrm{s(x,y)}| < 0.693$：

若$|x+y|>|x-y|$，則 $0\le \log\left(1+\frac{1}{e^{|x+y|}}\right)\le \log\left(1+\frac{1}{e^{|x-y|}}\right)$

$$|s(x,y)|=\log\left(1+\frac{1}{e^{|x+y|}}\right)-\log\left(1+\frac{1}{e^{|x-y|}}\right)$$

$$\le \log\left(1+\frac{1}{e^{|x+y|}}\right)\le \log\left(1+\frac{1}{e^{|x-y|}}\right)$$

$$\le \log\left(1+\frac{1}{e^{0}}\right)=\log 2\approx 0.693$$

若$|x+y|<|x-y|$，則 $0\le \log\left(1+\frac{1}{e^{|x-y|}}\right)\le \log\left(1+\frac{1}{e^{|x+y|}}\right)$

$$|s(x,y)|=\log\left(1+\frac{1}{e^{|x+y|}}\right)-\log\left(1+\frac{1}{e^{|x-y|}}\right)$$

$$\le \log\left(1+\frac{1}{e^{|x-y|}}\right)\le \log\left(1+\frac{1}{e^{|x+y|}}\right)$$

$$\le \log\left(1+\frac{1}{e^{0}}\right)=\log 2\approx 0.693$$

由引理 6 及$|s(x,y)|<0.693$，可以再取近似值：

$$L_1 \boxplus L_2 \approx \text{sign}(L_1)\text{sign}(L_2)\min(|L_1|,|L_2|)$$

$$\Rightarrow \text{LLR}(r_{ij})=\prod_{j'\epsilon V_i\backslash j}\text{sign}\left(\text{LLR}(q_{ij'})\right)\cdot \min_{j'\epsilon V_i\backslash j}(|\text{LLR}(q_{ij'})|)$$

第八章 模擬結果

8.1 模擬方式

在電腦上用 C 語言來實作訊息傳輸的過程，設定來模擬的是一組(3990,1995)的 LDPC codes。會先將穿送的訊息設置為 0，隨機產生雜訊，再將訊息加雜訊解碼產生的結果來計算錯誤率。

再過程中 AWGN 通道雜訊的模擬，我們必須先設定雜訊分佈的變異數。而通道品質是用雜訊比(signal to noise ratio, 雜訊比)定義為：

$$\frac{\mathrm{E_b}}{\mathrm{N_0}} = \frac{\mathrm{P}}{2\mathrm{R}\sigma^2}$$

其中$\mathrm{E_b}$代表每個位元的能量(bit energy)，編碼率$\mathrm{R} = \frac{\mathrm{k}}{\mathrm{n}}$，雜訊變異數$\sigma^2$，P為功率，在這裡我們假設$\mathrm{P} = 1$。而雜訊比通常以分貝為單位：

$$\mathrm{SNR} = 10\log_{10}\left(\frac{E_b}{N_0}\right)\ (db)$$

在模擬中，預設 SNR，再推出對應的變異數：

$$\mathrm{SNR} = 10\log_{10}\left(\frac{E_b}{N_0}\right)$$

$$\Rightarrow \frac{\mathrm{SNR}}{10} = \log_{10}\left(\frac{E_b}{N_0}\right)$$

$$\Rightarrow 10^{\frac{\mathrm{SNR}}{10}} = \frac{E_b}{N_0}$$

$$\Rightarrow \text{根據定義} \Rightarrow 10^{\frac{\mathrm{SNR}}{10}} = \frac{\mathrm{P}}{2\mathrm{R}\sigma^2}$$

$$\Rightarrow \mathrm{P} = 1 \Rightarrow 10^{-\frac{\mathrm{SNR}}{10}} = 2\mathrm{R}\sigma^2$$

$$\Rightarrow \mathrm{R} = \frac{\mathrm{k}}{\mathrm{n}} \Rightarrow \sigma^2 = \frac{1}{2}\frac{n}{k}10^{-\frac{\mathrm{SNR}}{10}}$$

預設的 SNR 為 1.3~2.3 以 0.1 為間格做模擬。

8.2 模擬觀察

有三組模擬的演算法，一組是機率領域 SPA，一組是對數領域 SPA，還有一組降低複雜度對數 SPA。針對 SNR 從 1.3~2.3，及疊代次數分別是 1、8、16、24、80 次，每一種都做 1000 次再合起來統計。圖表為錯誤率與 SNR 的關係，上下不同顏色的線表示疊代次數不同。其中 0 次是將未疊代直接判定的錯誤率放入做對比。

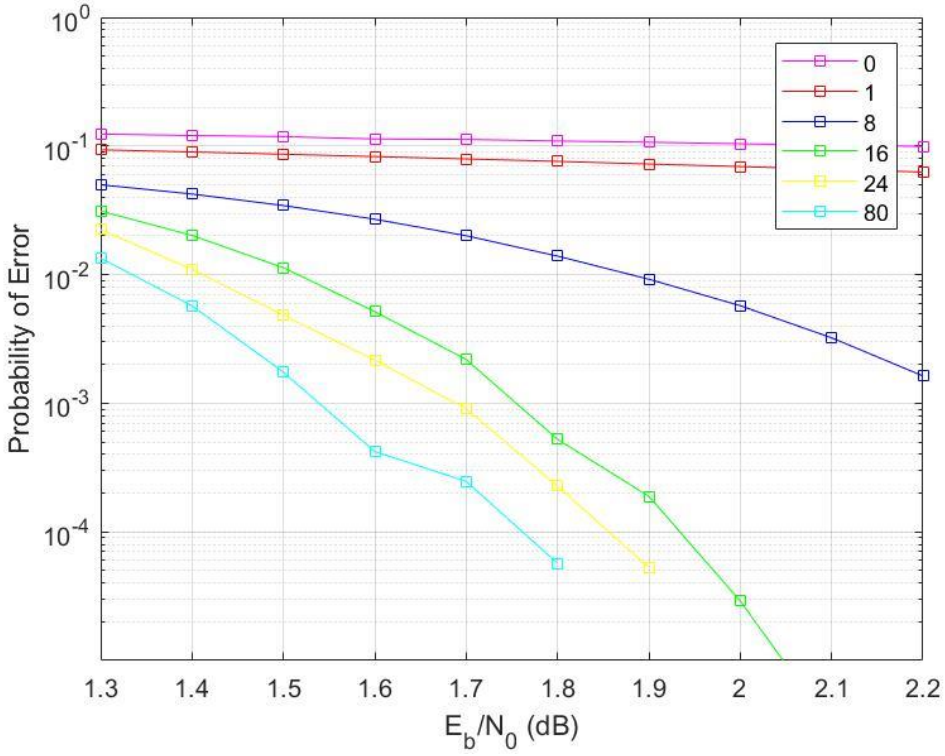


圖. 機率領域 SPA 的錯誤率-SNR 圖

從上圖可以看到在 1 次疊代後就有降低錯誤率的效果，隨著疊代次數增加，錯誤率也跟著快速減少，但是在通道環境差的情況下(SNR 低的情況)，錯誤率就沒辦法有效下降，因為會更容易產生在滿足同位檢查矩陣卻是錯誤的碼。

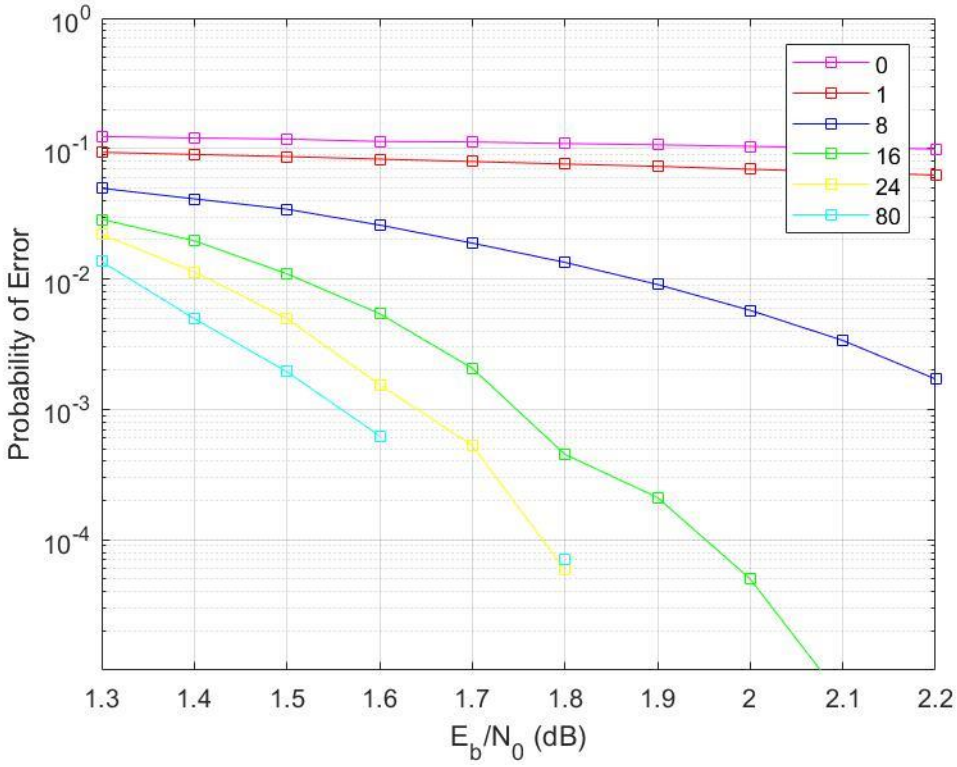


圖. 對數領域 SPA 的錯誤率-SNR 圖

從上圖可以發現對數領域 SPA 與機率領域 SPA 解碼錯誤率在相同條件的差異很小，幾乎是相同的，所以兩者差異只在計算的穩定度上。在對數領域 SPA 計算過程中有 $\phi(|\,\mathrm{LLR}(q_{ij'})|) = -\log\left(\tanh\frac{|\,\mathrm{LLR}(q_{ij'})|}{2}\right)$，當$\mathrm{LLR}(q_{ij'})$是0的時候，是沒有定義的，要注意會導致錯誤，而在 C 的函示是直接定義成0。

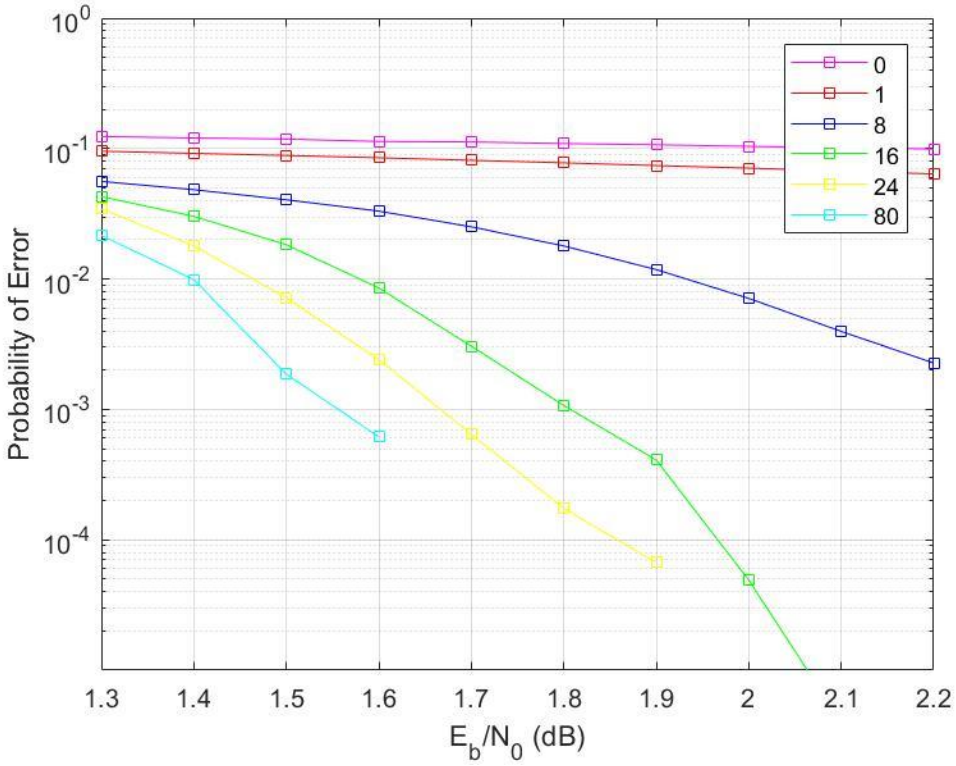


圖. 降低複雜度對數領域 SPA 的錯誤率-SNR 圖

在降低複雜度對數領域 SPA 的錯誤率數據中可以發現錯誤率大概都高出對數領域及機率領域 SPA 兩倍左右，可能是因為計算方式過程中不一樣的假設所造成的誤差。

結論

在這次專題中對於 LDPC 與疊代解碼演算法有了簡單的認識，主要是了解解碼演算法的推導過程及簡化，有機率領域、對數領域和簡化複雜度的對數領域演算法。也從實作中的驗證了解碼的效能。

參考資料

陳茂川, “低密度同位檢查碼的低複雜度疊代解碼演算法之介紹”, 中正大學 e-Thesys(98 學年度).

W. E. Ryan, “An introduction to LDPC codes,” August 2003.