國 立 交 通 大 學

電信工程學系

碩 士 論 文

軟性位元解碼在錯誤隱匿之研究

Softbit Decoding for Error Concealment over Noisy Channels

研 究 生：葉志杰

指導教授：張文輝 博士

中 華 民 國 九 十 四 年 六 月

軟性位元解碼在錯誤隱匿之研究

Softbit Decoding for Error Concealment over Noisy Channels

研 究 生：葉志杰　　**Student：Chih-Chieh Yeh**

指導教授：張文輝　　**Advisor：Wen-Whei Chang**

國立交通大學

電信工程學系碩士班

碩士論文

A Thesis

Submitted to Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of Requirements

for the Degree of

Master of Science

in

Communication Engineering

June 2005

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 四 年 六 月

軟性位元解碼在錯誤隱匿之研究

學生：葉志杰　　指導教授：張文輝 博士

國立交通大學電信工程學系碩士班

中文摘要

在數位通訊系統的設計，針對通道編碼器無法消除之殘餘位元錯誤作進一步的隱匿處理有其必要性。本論文研究一種結合硬性位元判定和其可靠度的軟性位元解碼演算法，其關鍵在利用量化索引的事前消息且使用 BCJR 軟性輸出解碼演算法所提供的位元可靠消息。傳統的 BCJR 演算法以位元層級為基礎，其目標為計算每一個傳送位元的事後機率。為了改善錯誤隱匿功能，我們修改軟性輸出通道解碼的 BCJR 演算法，主要特色是同時利用位元層級和索引層級的相關性。自迴歸訊號源的模擬結果指出，基於索引的 BCJR 演算法和索引的一階事前消息的使用，對於軟性位元解碼有最好的錯誤隱匿表現。

Softbit Decoding for Error Concealment over Noisy Channels

Student：Chih-Chieh Yeh Advisor：Dr. Wen-Whei Chang

Institute of Communication Engineering

National Chiao Tung University

Abstract

In digital communication over noisy channels there is a need for concealment of residual bit errors which have not been eliminated by standard channel decoding. A soft-bit decoding scheme based on joint knowledge of a hard bit and its reliability is studied and applied to quantization of autoregressive sources over AWGN channels. This approach exploits a priori knowledge about the quantizer's indexes and requires bit reliability information provided by the soft-output channel decoder using the BCJR algorithm. Conventional BCJR algorithm aims to compute the a posteriori probability of each transmitted bit and hence is based on the bit-level. For further improvement, we propose a modified BCJR algorithm which exploits bit-level correlation as well as index-level correlation in the soft-output channel decoding process. Simulation results indicate that for the soft-bit decoding, the combined use of a modified BCJR algorithm and the index's first-order a priori knowledge performs the best for error concealment under noisy channel conditions.

誌謝

本篇論文的完成，首先要感謝指導教授張文輝老師，由於老師的悉心指導，讓我了解到做研究的態度與方法。另外也要感謝實驗室的學長、同學以及學弟們，在課業或研究上協助我解決問題，同時，也要感謝我的朋友們，一路伴隨我走過研究所。最後，僅將此論文獻給我親愛的父母與家人。

目錄

中文摘要.....i

英文摘要.....ii

致謝.....iii

目錄.....iv

圖目錄.....vii

第一章 緒論.....1

1.1 研究動機與方向.....1

1.2 章節概要.....2

第二章 軟性位元解碼.....3

2.1 硬性位元解碼演算法.....3

2.2 軟性位元解碼演算法.....5

第三章 基於位元的軟性輸出通道解碼.....13

3.1 迴旋編碼器的架構.....14

3.2 以位元為基礎的 BCJR 演算法.....15

3.2.1 前向機率的推導.....18

3.2.2 後向機率的推導.....20

3.2.3 機率γ的推導.....22

3.2.4 對數相似比率的計算.....24

第四章 基於索引的軟性輸出通道解碼.....26

4.1 基於索引的迴旋編碼器架構.....26

4.2 基於索引的 BCJR 演算法.....28

4.3 前向和後向機率的初始條件設定.....32

第五章 實驗模擬與結果分析.....36

5.1 基於位元的軟性輸出通道解碼系統之模擬.....36

5.1.1 系統模擬之步驟說明.....36

5.1.2 結果分析.....40

5.2 基於索引的軟性輸出通道解碼系統架之模擬.....40

5.2.1 系統模擬之步驟說明.....40

5.2.2 結果分析.....43

5.3 基於位元與基於索引之系統綜合結果分析和比較.....43

第六章 結論與未來展望.....46

6.1 結論.....46

6.2 未來展望.....47

參考文獻.....48

圖目錄

圖 2.1 硬性位元解碼系統架構.....5

圖 2.2 軟性位元解碼系統架構.....6

圖 3.1 迴旋編碼器架構.....15

圖 3.2 前向機率計算的格子架構.....20

圖 3.3 後向機率計算的格子架構.....22

圖 4.1 基於索引的迴旋編碼器架構.....28

圖 5.1 基於位元的軟性位元解碼之系統模擬.....39

圖 5.2 基於索引的軟性位元解碼之系統模擬.....42

圖 5.3 基於位元與索引系統之比較.....44

圖 5.4 結合 AK1 與不同解碼之比較.....45

第一章 緒論

1.1 研究動機與方向

在典型的數位通訊系統裡，傳送的訊號源經通道到接收端，通常都會遭受到通道環境雜訊等諸多效應的干擾，致使接收到的訊號無法判別重建出原始傳送訊號的樣貌。所以我們必須設法改進通訊系統，以期增強其對抗通道雜訊之能力。

又在傳統的數位通訊系統裡，通常都是將訊源編碼（source coding）和通道編碼（channel coding）兩個系統區塊分開考量和獨立設計，此設計概念乃源自於沈農（Shannon）的消息理論[1]。此理論假設在設計個別的編碼架構時，前提假設另一個編碼器是最佳化設計，但這不符合實際的通訊環境。訊源編碼器在編碼處理後，無法完全去除輸出訊號間之關聯性，即輸出訊號之間彼此不是獨立同等分佈（i.i.d.），其間存在某種型式的分佈或隱含有記憶性，而通道編碼的輸出亦然。因此，若能妥善利用訊號源在編碼後之殘餘冗息（residual redundancy），應能有效提升系統效能，進而對抗通道雜訊之干擾。有鑑於此，合併訊源通道解碼（joint source-channel decoding）[2]相關研究之目的是，在接收端將訊源解碼器和通道解碼器之效應一併納入架構設計的考量。

1.2 章節概要

第二章介紹了軟性位元解碼演算法。第三章描述了基於位元的軟性輸出通道解碼演算法的架構。第四章則為基於索引的軟性輸出通道解碼演算法。第五章為實驗模擬，模擬前二章所提及之演算法架構，並且為基於此兩架構下之模擬結果作效能評估與比較。第六章為結論與未來展望。

第二章 軟性位元解碼

在本章中，我們介紹了一個近年來發展用於對語音在傳送時做錯誤隱匿（error concealment）[3]的方法，稱為軟性位元解碼演算法（soft-bit decoding）[4]。有別於傳統的硬性位元解碼（hard-bit decoding），其主要的差異點在於軟性位元解碼不僅須執行位元為 0 或 1 的硬性判定，還得知道該位元判定的可靠度（reliability），亦即位元錯誤機率的正確估計。結合這兩項資訊，並利用事前針對大量訓練語料分析所得的殘餘冗息（residual redundancy），實驗證實可以有效對抗傳輸錯誤取得語音參數的最佳估測結果。

以下，我們將首先在第一節裡介紹硬性位元解碼演算法，接著在第二節介紹軟性位元解碼演算法，並說明其演算法之流程架構。

2.1 硬性位元解碼演算法

考慮傳送一個特定參數$v_t \in \mathrm{R}$至一個雜訊通道，其系統架構圖如圖 2.1 所示。首先將此參數v_t進行M位元的量化處理而得$Q(v_t)=c_{x_t}$，其中x_t為一索引（index）且屬於一組有限的整數集合$J=\{0,1,...,2^M-1\}$。經由位元對映轉換（bit mapping, BM），針對每一個整數索引值x_t指派一個由M個位元所構成的位元組合：

$$\underline{X}_t \triangleq \left\{x_{t,0}, x_{t,1}, ..., x_{t,M-1}\right\} \text{。} \qquad (2.1)$$

為了標記符號的簡化，我們使用另一個變數u_k來表示訊息位元（information bit）$u_k = x_{t,m}$，$m = 0,1,...,M-1$，其中$k = tM + m$是以位元方式表示的時間索引(bit-wise time index)。在時間為k時，將單一訊息位元u_k送進一個通道編碼器（channel encoder)，作編碼處理後再輸出一個N_l位元組合$\underline{Y}_k \triangleq \left\{y_{k,0}, y_{k,1}, ..., y_{k,N_l-1}\right\}$，其中$1/N_l$為通道編碼率（channel coding rate)。若考慮經由雜訊干擾的通道傳送一群N_c個訊息位元，則接收到的實數序列為$\underline{\hat{Y}}_k \triangleq \left\{\hat{y}_{k,0}, \hat{y}_{k,1}, ..., \hat{y}_{k,N_l-1}\right\}$，$k = 0,1,..., N_c - 1$。對個別位元而言，

$$\hat{y}_{k,l} = y_{k,l} + n_{k,l} \text{，} \qquad (2.2)$$

其中的$n_{k,l}$為零平均值，且變異數為$\sigma^2 = N_0/(2E_s)$的高斯雜訊取樣，E_s為每一編碼符號的能量，而N_0為通道的單邊雜訊功率頻譜密度。在接收端，硬性位元解碼的工作流程如下：經過通道解碼器（channel decoder）處理後，所得的位元組合$\underline{\tilde{X}}_t \triangleq \left\{\tilde{x}_{t,0}, \tilde{x}_{t,1}, ..., \tilde{x}_{t,M-1}\right\}$執行反向位元對映轉換（inverse bit mapping, BM^{-1}）找出相對應的索引$\tilde{x}_t$。最後，再經過碼書查詢(codebook lookup)而決定出被重建的參數$\hat{v}_t$。

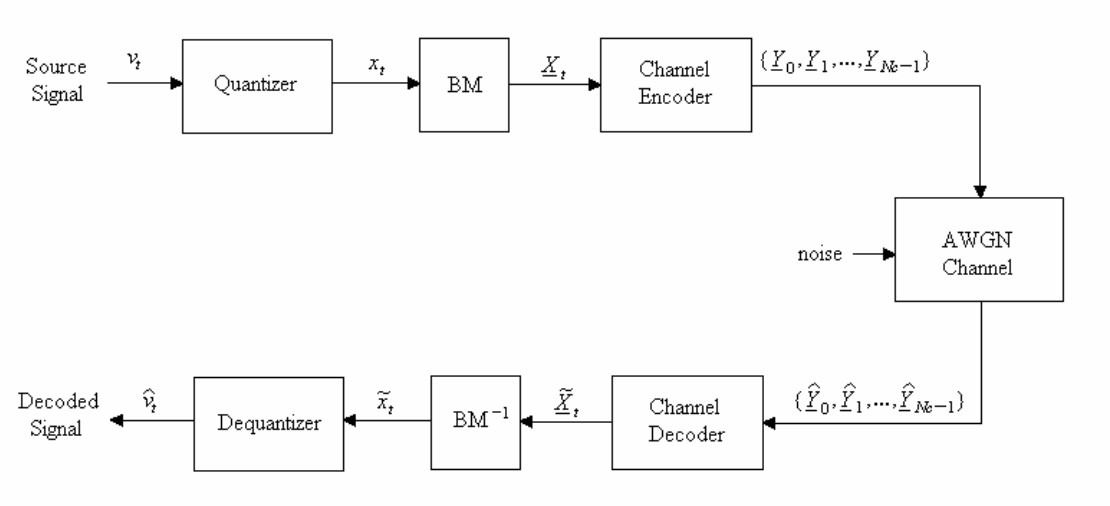

圖 2.1 硬性位元解碼系統架構

2.2 軟性位元解碼演算法

軟性位元具有雙重的涵意，除了對接收訊號位元 $x_{t,m}$ 作硬性判定而得一位元 $\tilde{x}_{t,m} = i \in \{-1,1\}$ 之外，作此判定的位元錯誤機率 $P_e\left(x_{t,m}\right)$ 也要被考慮進去。而其中的 $P_e\left(x_{t,m}\right)$ 代表接收到的訊號受到通道特性和通道編碼的影響，而致使 $x_{t,m}$ 是否能被正確解碼的機率。換言之，所謂的軟性位元 $\left\{\tilde{x}_{t,m}, P_e\left(x_{t,m}\right)\right\}$ [5]，結合了接收訊號的硬性判定（hard decision）和關於此判定的可靠訊息（reliability information）。軟性位元解碼的整體系統架構，如圖 2.2 所示，主要包括下列五項執行步驟。

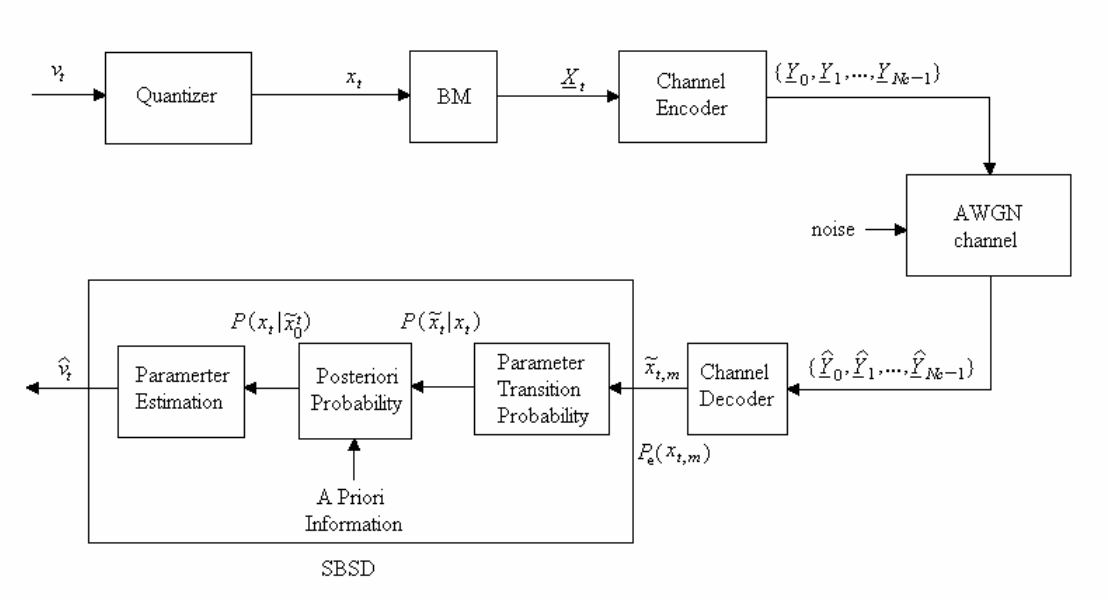

圖 2.2 軟性位元解碼系統架構

【1】可靠訊息的計算

在傳送端使用通道編碼器的情況下，為了得到每一個解碼後位元的可靠訊息，我們需要採用有軟性輸出的通道解碼器（soft-output channel decoder）。在軟性輸出通道解碼器[6]的輸出端，一個傳送位元$x_{t,m}$的對數相似比率（log-likelihood ratio, LLR）通常表示為：

$$L_0\left(x_{t,m}\right)=\log\frac{P\left(x_{t,m}=1|\hat{\underline{Y}}_0^{N_c-1}\right)}{P\left(x_{t,m}=-1|\hat{\underline{Y}}_0^{N_c-1}\right)} \qquad (2.3)$$

其中$P\left(x_{t,m}=i|\hat{\underline{Y}}_0^{N_c-1}\right)$，$i\in\{1,-1\}$為訊息位元$x_{t,m}$的事後機率（a posteriori probabilities, APPs），而接收端輸入至通道解碼器的序列則為$\hat{\underline{Y}}_0^{N_c-1}=\left\{\hat{\underline{Y}}_0,\hat{\underline{Y}}_1,...,\hat{\underline{Y}}_{N_c-1}\right\}$。當使用迴旋碼（convolutional code）於通道編碼器時，對數相似比率$L_0\left(x_{t,m}\right)$及解碼位元的事後機

率的推導將於下個章節中詳述。

取得$L_0\left(x_{t,m}\right)$值之後，傳送位元$x_{t,m}$的硬性判定為：

$$\tilde{x}_{t,m}=\text{sign}\left[L_0\left(x_{t,m}\right)\right] \tag{2.4}$$

而其相對應的可靠訊息為：

$$P_e\left(x_{t,m}\right)=\begin{cases}P\left(x_{t,m}=1|\hat{\underline{Y}}_0^{N_c-1}\right) & \text{if } L_0\left(x_{t,m}\right)<0\\ P\left(x_{t,m}=-1|\hat{\underline{Y}}_0^{N_c-1}\right) & \text{if } L_0\left(x_{t,m}\right)>0\end{cases} \tag{2.5}$$

由（2.3）式，我們可以推導出：

$$P\left(x_{t,m}=-1|\hat{\underline{Y}}_0^{N_c-1}\right)=\frac{1}{1+\exp\left[L_0\left(x_{t,m}\right)\right]} \tag{2.6}$$

和

$$\begin{aligned}P\left(x_{t,m}=1|\hat{\underline{Y}}_0^{N_c-1}\right)&=1-P\left(x_{t,m}=-1|\hat{\underline{Y}}_0^{N_c-1}\right)\\&=\frac{\exp\left[L_0\left(x_{t,m}\right)\right]}{1+\exp\left[L_0\left(x_{t,m}\right)\right]}\\&=\frac{1}{1+\exp\left[-L_0\left(x_{t,m}\right)\right]}\end{aligned} \tag{2.7}$$

比較（2.6）、（2.7）、及（2.5）式，可靠訊息可表示為：

$$P_e\left(x_{t,m}\right)=\frac{1}{1+\exp\left|L_0\left(x_{t,m}\right)\right|} \tag{2.8}$$

【2】索引的通道移轉機率之計算

在給定一傳送位元$x_{t,m}$情況下，解碼所得的硬性位元為$\tilde{x}_{t,m}$的通道移轉機率（channel transition probability）為：

$$P\left(\tilde{x}_{t,m}\mid x_{t,m}\right)=\begin{cases}1-P_e\left(x_{t,m}\right) & \text{if } \tilde{x}_{t,m}=x_{t,m}\\ P_e\left(x_{t,m}\right) & \text{if } \tilde{x}_{t,m}\neq x_{t,m}\end{cases} \qquad (2.9)$$

進一步假設通道為無記憶性（memoryless），則索引x_t的通道移轉機率將為：

$$P\left(\tilde{x}_t\mid x_t\right)=\prod_{m=0}^{M-1}P\left(\tilde{x}_{t,m}\mid x_{t,m}\right)\text{，}\ x_t\in\left\{0,1,...,2^M-1\right\} \qquad (2.10)$$

此項提供了2^M種可能的傳送索引x_t最後經解碼判定為索引$\tilde{x}_t$的機率。

【3】事前消息的計算

在一般的情況下，通常會把量化輸出的索引序列模擬成一個N階馬可夫程序（Nth-order Markov process），即

$$P\left(x_t\mid x_0^{t-1}\right)=P\left(x_t\mid x_{t-N}^{t-1}\right) \qquad (2.11)$$

其中$x_0^{t-1}\triangleq\left\{x_0,x_1,...,x_{t-1}\right\}$。由此可知，量化索引序列若視為$N$階馬可夫程序，意謂著現在發生的事件僅與過去剛發生的N個事件有關聯性，亦即此序列在時間上有可回溯N個時間點的記憶特性。

為了找出適當的馬可夫階數，我們通常會先行量測索引序列的相關機率，如$P\left(x_t\right)$、$P\left(x_t\mid x_{t-1}\right)$，或者是更高階的條件機率。這些機率值皆以訓練（training）方式求得，主要是將語音資料庫裡的大量語

料作量化處理，然後計算每一量化索引的發生次數或相鄰兩量化索引共同發生的次數，進而求得其條件機率。

我們稱$P(x_t)$為零階事前消息（0*th* order a priori knowledge, AK0），因其在統計上的描述為零階的馬可夫程序，亦即表示為一個沒有記憶性的程序。若假設每一個索引的發生機率相同，$P(x_t)=1/2^M$，則稱之為無事前消息（no a priori knowledge, NAK）。除此之外，我們稱$P(x_t|x_{t-1})$為一階事前消息（first-order a priori knowledge, AK1），因其代表了一階的馬可夫程序。至於馬可夫程序階數的最佳設定，必須參考三項要點：1)量化索引的殘餘冗息，2)運算複雜度，及3)效能和運算複雜度之間的衡量取捨。

【4】量化索引事後機率的計算

在接收端，首要任務是準確估測每一個量化索引x_t的事後機率（a posteriori probabilities）。而其計算方法，又會因為索引序列的馬可夫階數作不同假設而有明顯差異。

(A) NAK 的計算方法：

如果沒有索引序列的事前消息可用，就必須假設量化器的輸出索引互不相關（uncorrelated）且機率皆為均等（equally likely）。在此情形下，就只能利用通道的相關訊息，因此x_t的事後機率為：

$$P(x_t \mid \tilde{x}_t) = C \cdot P(\tilde{x}_t \mid x_t) \qquad (2.12)$$

其中C為正規化常數：

$$C = \frac{1}{\sum_{x_t \in J} P(\tilde{x}_t \mid x_t)}$$

其中$J = \{0,1,...,2^M - 1\}$。常數C是用來正規化事後機率，而使得$\sum_{x_t \in J} P(x_t \mid \tilde{x}_t) = 1$。特別強調的是，在一般實際的編碼架構下，量化輸出索引為機率均等的假設，通常是不成立的。舉例而言，廣為使用的Lloyd-Max量化器，在每一個量化間隔內，會產生相同的量化錯誤變異數（quantization error variance），但並不會產生出均等的機率值$P(x_t)$。

(B) AKO的計算方法：

如果量化索引序列提供零階事前消息，再配合使用貝氏定理（Bayes rule），即可推導其事後機率為：

$$P(x_t \mid \tilde{x}_t) = C \cdot P(\tilde{x}_t \mid x_t) \cdot P(x_t) \qquad (2.13)$$

其中正規化常數C為：

$$C = \frac{1}{\sum_{x_t \in J} P(\tilde{x}_t \mid x_t) \cdot P(x_t)}$$

若所有量化輸出索引的機率均等，即$P(x_t) = 2^{-M}$，那麼（2.13）式將簡化為（2.12）式，使用AKO與NAK的方法完全相同。

(C) AK1的計算方法：

在上述 AK0 的方法中，僅使用到個別索引的事前機率$P(x_t)$，這是因為我們假設相鄰兩索引之間為互相獨立。但若是索引間仍存在有一階的殘餘相關性，便可擴充我們的事後機率計算，將此一階的索引事前訊息也納入考量之中。

在接收端，我們可利用的最大訊息量來自直至目前為止已完成的通道解碼器輸出序列$\tilde{x}_0,\tilde{x}_1,...,\tilde{x}_t$。因此利用大量語料訓練而得的事前機率$P(x_t \mid x_{t-1})$和已接收到的索引序列$\tilde{x}_0^t \triangleq \{\tilde{x}_0,\tilde{x}_1,...,\tilde{x}_t\}$，我們可以求得個別索引的事後機率，並以遞迴的形式呈現如下：

$$P(x_t \mid \tilde{x}_0^t) = C \cdot P(\tilde{x}_t \mid x_t) \cdot \sum_{x_{t-1} \in J} P(x_t \mid x_{t-1}) \cdot P(x_{t-1} \mid \tilde{x}_0^{t-1}) \quad (2.14)$$

其中正規化常數C為：

$$C = \frac{1}{\sum_{x_t \in J} P(\tilde{x}_t \mid x_t) \cdot \sum_{x_{t-1} \in J} P(x_t \mid x_{t-1}) \cdot P(x_{t-1} \mid \tilde{x}_0^{t-1})}$$

倘若相鄰索引間沒有殘餘的相關性，$P(x_t \mid x_{t-1}) = P(x_t)$，則使用 AK1 與使用 AK0 的結果相同。另外，值得一提的是，(2.14) 在計算上並不需要任何時間上的延遲。

【5】參數估測

若能順利依次完成上述的幾個步驟，取得每一個傳送索引的事後機率$P(x_t \mid \tilde{x}_t)$或$P(x_t \mid \tilde{x}_0^t)$，$x_t \in J$，最後一個步驟就是根據事後機率進行參數估測。就一般語音參數而言，大多使用最小均方錯誤準則

（minimum mean square error criterion）來估測參數。依其估測準則，重建後的參數$\hat{v}_t$如下所示：

$$\hat{v}_t = \sum_{x_t \in J} c_{x_t} \cdot P\left(x_t \mid \tilde{x}_0^t\right) \qquad (2.15)$$

其中的c_{x_t}為索引x_t所對應的量化參數值。

第三章 基於位元的軟性輸出通道解碼

為了使通訊系統在傳送的過程中，擁有錯誤保護的功能，我們加入了通道編碼。在此，加入的通道編碼為廣受一般所使用的迴旋碼。伴隨著迴旋碼的使用而在通道解碼器擔任相對應解碼工作的通常是 Viterbi 演算法，但如上一章所述，為了使用軟性位元解碼，通道解碼器必須提供解每一解碼後位元的硬性判定和其可靠訊息。不幸的是，在傳統的 Viterbi 演算法架構下，通道解碼器僅能執行硬性判定而提供最大可能傳送序列（the most probable transmitted sequence）的訊息，無法供給軟性位元解碼所需的可靠訊息。為克服這個問題，於是有軟性輸出 Viterbi 演算法（soft-output Viterbi algorithm, SOVA）[7]被提出。雖然藉著其改善，可以提供每一解碼後位元的軟性輸出，但其仍屬於次佳（sub-optimal）的估測，因為 SOVA 不是針對每一解碼後位元的可靠訊息作最佳的估測。而就我們所知的，BCJR 演算法[6][8]可以提供每一解碼後位元的可靠訊息，且它會針對最小化位元錯誤機率作最佳估測。因此在軟性位元解碼演算法裡，我們用 BCJR 演算法當作是迴旋碼的解碼演算法。

在本章中，我們將先大略簡介迴旋碼通道編碼器的架構，接著詳述以位元為基礎的 BCJR 演算法（bit-based BCJR algorithm）架構

及推導。

3.1 迴旋編碼器的架構

考慮一個限制長度（constraint length）為M_c+1而暫存記憶體為M_c的迴旋碼編碼器，假設在時間k，編碼器的輸入為消息位元（information bit）u_k，而輸出為編碼位元組（coded bits）$\underline{Y}_k \triangleq \left\{y_{k,0}, y_{k,1}, ..., y_{k,N_l-1}\right\}$，其中$1/N_l$為編碼率（coding rate）。對個別位元而言，$y_{k,l}, l=0,1,...,N_l-1$由下式產生：

$$y_{k,l} = \sum_{i=0}^{M_c} g_i^l u_{k-i} \mod 2, \quad g_i^l \in \{0,1\} \qquad (3.1)$$

其中編碼產生器多項式（encoder generator polynomials）定義為：

$$G_l(D) = g_0^l + Dg_1^l + D^2 g_1^l + \ldots + D^{M_c} g_{M_c}^l, \quad l=0,1,\ldots,N_l-1 \qquad (3.2)$$

一個典型的迴旋碼編碼器的架構圖如圖 3.1 所示，圖中架構的限制長度為5和編碼率為$1/2$，而產生器多項式為

$$G_1(D) = 1 + D^2 + D^3 + D^4 \qquad (3.3)$$

$$G_2(D) = 1 + D^1 + D^4 \qquad (3.4)$$

編碼器的狀態S_k由移位暫存器（shift register）的最近M_c個輸入位元所決定：

$$S_k = \sum_{i=0}^{M_c-1} 2^i u_{k-i-1} \qquad (3.5)$$

在時間為零時，初始狀態$S_0=0$，當消息位元傳送結束時，必須多輸入額外M_c個為0的尾巴位元（tail bits）進入編碼器，以迫使編碼器的狀態回歸至0。

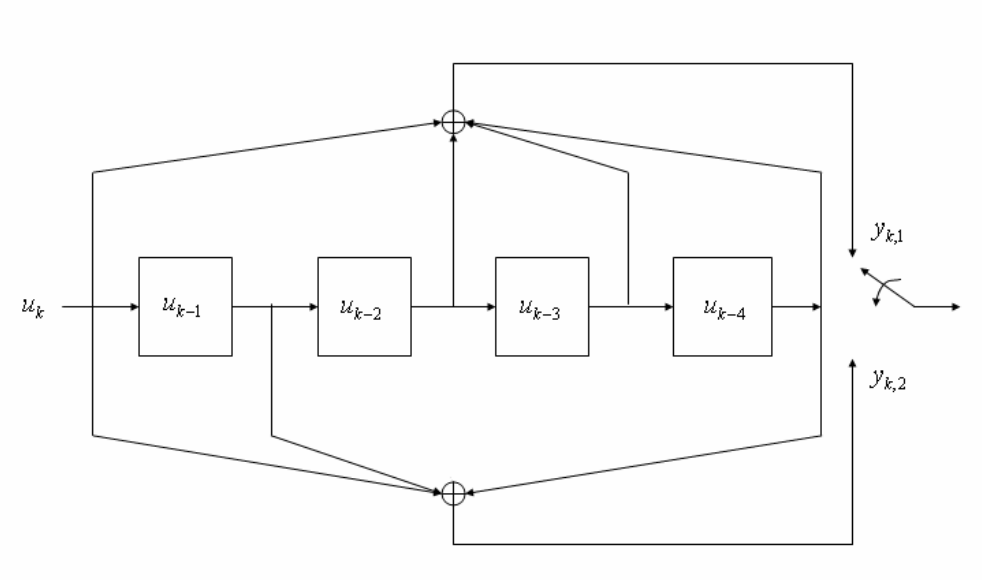

圖 3.1 迴旋編碼器架構

3.2 以位元為基礎的 BCJR 演算法

假設通訊系統裡的通道編碼器之架構如上一節所提及，並且假設在每次的傳送以N_c個消息位元為一個區塊，如此，接收到的實數序列為$\underline{\hat{Y}}_k \triangleq \left\{\hat{y}_{k,0},\hat{y}_{k,1},...,\hat{y}_{k,N_l-1}\right\}$，$k=0,1,...,N_c-1$。對個別位元而言，

$$\hat{y}_{k,l}=y_{k,l}+n_{k,l} \qquad (3.6)$$

其中的$n_{k,l}$為零平均值，且變異數為$\sigma^2=N_0/(2E_s)$的高斯雜訊取樣，E_s為每一編碼符號的能量，而N_0為通道的單邊雜訊功率頻譜密度。

在上一章中，為了使用軟性位元解碼，我們必須求得硬性判定的

可靠訊息：

$$P_e\left(x_{t,m}\right)=\frac{1}{1+\exp\left|L_0\left(x_{t,m}\right)\right|} \qquad (3.7)$$

而為了獲得這個可靠訊息，同樣地，我們必須先求得上式中的對數相似比率：

$$L_0\left(x_{t,m}\right)=\log\frac{P\left(x_{t,m}=1\,|\,\hat{\underline{Y}}_0^{N_c-1}\right)}{P\left(x_{t,m}=-1\,|\,\hat{\underline{Y}}_0^{N_c-1}\right)} \qquad (3.8)$$

在此，我們便用以位元為基礎的 BCJR 演算法估測出上式中的事後機率 $P\left(x_{t,m}=i\,|\,\hat{\underline{Y}}_0^{N_c-1}\right)=P\left(u_k=i\,|\,\hat{\underline{Y}}_0^{N_c-1}\right)$，其中 $k=tM+m$ 而 $i\in\{-1,+1\}$。

我們可由上一節迴旋編碼器的架構圖中，觀察得知編碼器的輸出位元組合會因當時輸入編碼器的消息位元和當時編碼器所儲存的狀態所共同決定。也就是說，編碼輸出位元組合會因編碼器裡暫存的狀態而產生關聯性，因此在時間點 k，計算通道解碼器的軟性輸出時，我們將通道編碼器的狀態 s_k 也納入考量，那麼對數相似比率 $L_0\left(u_k\right)$ 可被寫成如下所示：

$$L_0(u_k)=\log\frac{P\left(u_k=1|\hat{\underline{Y}}_0^{N_c-1}\right)}{P\left(u_k=-1|\hat{\underline{Y}}_0^{N_c-1}\right)}$$

$$=\log\frac{\sum_{s_k}P\left(u_k=1,s_k|\hat{\underline{Y}}_0^{N_c-1}\right)}{\sum_{s_k}P\left(u_k=-1,s_k|\hat{\underline{Y}}_0^{N_c-1}\right)}$$

$$=\log\frac{\sum_{s_k}\lambda_k^1(s_k)}{\sum_{s_k}\lambda_k^{-1}(s_k)} \qquad (3.9)$$

其中$\lambda_k^i(s_k)$為結合機率（joint probability），定義為

$$\lambda_k^i(s_k)=P\left(u_k=i,S_k=s_k|\hat{\underline{Y}}_0^{N_c-1}\right),\ i\in\{1,-1\} \qquad (3.10)$$

利用貝氏定理（Bayes's rule），$\lambda_k^i(s_k)$可以被表示為

$$\lambda_k^i(s_k)=P\left(u_k=i,S_k=s_k|\hat{\underline{Y}}_0^{N_c-1}\right)$$

$$=\frac{P\left(u_k=i,S_k=s_k,\hat{\underline{Y}}_0^{N_c-1}\right)}{\sum_i\sum_{s_k}P\left(u_k=i,S_k=s_k,\hat{\underline{Y}}_0^{N_c-1}\right)}$$

$$=\frac{P\left(\hat{\underline{Y}}_{k+1}^{N_c-1},\hat{\underline{Y}}_0^{k},u_k=i,S_k=s_k\right)}{\sum_i\sum_{s_k}P\left(u_k=i,S_k=s_k,\hat{\underline{Y}}_0^{N_c-1}\right)}$$

$$=\frac{P\left(\hat{\underline{Y}}_{k+1}^{N_c-1}|u_k=i,S_k=s_k,\hat{\underline{Y}}_0^{k}\right)P\left(u_k=i,S_k=s_k,\hat{\underline{Y}}_0^{k}\right)}{\sum_i\sum_{s_k}P\left(u_k=i,S_k=s_k,\hat{\underline{Y}}_0^{N_c-1}\right)}$$

$$=\frac{\alpha_k^i(s_k)\beta_k^i(s_k)}{\sum_i\sum_{s_k}\alpha_k^i(s_k)\beta_k^i(s_k)} \qquad (3.11)$$

其中

$$\alpha_k^i(s_k) = P\left(u_k = i, S_k = s_k, \hat{\underline{Y}}_0^k\right) \qquad (3.12)$$

$$\beta_k^i(s_k) = P\left(\hat{\underline{Y}}_{k+1}^{N_c-1} \mid u_k = i, S_k = s_k\right) \qquad (3.13)$$

分別被稱作前向（forward）和後向（backward）機率。因此，BCJR 演算法也被稱作是前向-後向演算法（forward-backward algorithm)。更重要的是前向和後向機率可以各自被推導成遞迴的型式，以便於計算及降低運算量，其型式如下所示：

$$\alpha_k^i(s_k) = \sum_{s_{k-1}} \sum_{j\in\{-1,1\}} \alpha_{k-1}^j(s_{k-1})\gamma_{i,j}\left(\hat{\underline{Y}}_k, s_{k-1}, s_k\right) \qquad (3.14)$$

和

$$\beta_k^i(s_k) = \sum_{s_{k+1}} \sum_{j\in\{-1,1\}} \beta_{k+1}^j(s_{k+1})\gamma_{j,i}\left(\hat{\underline{Y}}_{k+1}, s_{k+1}, s_k\right) \qquad (3.15)$$

在上式中的α、β、和γ，其更詳細的推導將於接下來的三個次小節中說明。

3.2.1 前向機率的推導

前向機率的定義為

$$\alpha_k^i(s_k) = P\left(u_k = i, S_k = s_k, \hat{\underline{Y}}_0^k\right),\ i = 1, -1 \qquad (3.16)$$

其可展開為

$$\alpha_k^i(s_k) = P\left(u_k = i, S_k = s_k, \underline{\hat{Y}}_0^{k-1}, \underline{\hat{Y}}_k\right)$$

$$= \sum_{s_{k-1}} \sum_{j\in\{-1,1\}} P\left(u_k = i, u_{k-1} = j, S_k = s_k, S_{k-1} = s_{k-1}, \underline{\hat{Y}}_0^{k-1}, \underline{\hat{Y}}_k\right)$$

$$= \sum_{s_{k-1}} \sum_{j\in\{-1,1\}} P\left(u_k = i, S_k = s_k, \underline{\hat{Y}}_k \mid u_{k-1} = j, S_{k-1} = s_{k-1}, \underline{\hat{Y}}_0^{k-1}\right)$$

$$\times P\left(u_{k-1} = j, S_{k-1} = s_{k-1}, \underline{\hat{Y}}_0^{k-1}\right) \qquad (3.17)$$

因為從s_{k-1}到s_k的轉移路徑完全由u_{k-1}和s_{k-1}決定，而和接收到的序列$\underline{\hat{Y}}_0^{k-1}$沒有任何的關係，所以

$$P\left(u_k = i, S_k = s_k, \underline{\hat{Y}}_k \mid u_{k-1} = j, S_{k-1} = s_{k-1}, \underline{\hat{Y}}_0^{k-1}\right)$$

$$= P\left(u_k = i, S_k = s_k, \underline{\hat{Y}}_k \mid u_{k-1} = j, S_{k-1} = s_{k-1}\right)$$

$$\triangleq \gamma_{i,j}\left(\underline{\hat{Y}}_k, s_{k-1}, s_k\right) \qquad (3.18)$$

除此之外，

$$P\left(u_{k-1} = j, S_{k-1} = s_{k-1}, \underline{\hat{Y}}_0^{k-1}\right) = \alpha_{k-1}^j(s_{k-1}) \qquad (3.19)$$

所以（3.17）式可簡化成以下的前向遞迴計算型式：

$$\alpha_k^i(s_k) = \sum_{s_{k-1}} \sum_{j\in\{-1,1\}} \alpha_{k-1}^j(s_{k-1})\gamma_{i,j}\left(\underline{\hat{Y}}_k, s_{k-1}, s_k\right) \qquad (3.20)$$

在編碼的過程中，通道編碼器的初始狀態被假設為$s_0 = 0$，所以α的初始值被設為：

$$\alpha_0^i(s_0) = P\left(u_0 = i, S_0 = s_0, \underline{\hat{Y}}_0\right)$$

$$= \begin{cases} P\left(\underline{\hat{Y}}_0 \mid u_0 = i, S_0 = 0\right)P\left(u_0 = i\right) & \text{if } s_0 = 0 \\ 0 & \text{if } s_0 \neq 0 \end{cases} \qquad (3.21)$$

前向機率遞迴計算所對應的格子架構（trellis）如圖 3.2 所示。

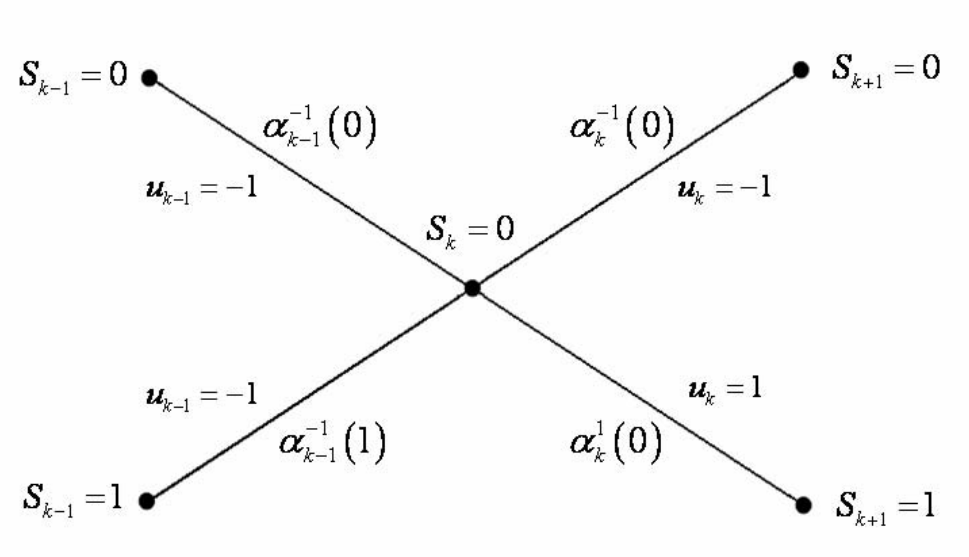

圖 3.2 前向機率計算的格子架構

3.2.2 後向機率的推導

後向機率的定義為

$$\beta_k^i(s_k)=P\left(\underline{\hat{Y}}_{k+1}^{N_c-1}\,|\,u_k=i,S_k=s_k\right) \qquad (3.22)$$

其遞迴計算的推導和$\alpha_k^i(s_k)$相似，但它必須在整個序列接收完全後，才可以開始計算。(3.22) 式可寫成

$$\begin{aligned}\beta_k^i(s_k)&=P\left(\underline{\hat{Y}}_{k+2}^{N_c-1},\underline{\hat{Y}}_{k+1}\,|\,u_k=i,S_k=s_k\right)\\&=\sum_{s_{k+1}}\sum_{j\in\{-1,1\}}P\left(u_{k+1}=j,S_{k+1}=s_{k+1},\underline{\hat{Y}}_{k+2}^{N_c-1},\underline{\hat{Y}}_{k+1}\,|\,u_k=i,S_k=s_k\right)\\&=\sum_{s_{k+1}}\sum_{j\in\{-1,1\}}P\left(\underline{\hat{Y}}_{k+2}^{N_c-1}\,|\,u_{k+1}=j,S_{k+1}=s_{k+1},\underline{\hat{Y}}_{k+1},u_k=i,S_k=s_k\right)\\&\times P\left(u_{k+1}=j,S_{k+1}=s_{k+1},\underline{\hat{Y}}_{k+1}\,|\,u_k=i,S_k=s_k\right)\end{aligned} \qquad (3.23)$$

因為在時間$k+1$之後的路徑僅視u_{k+1}和s_{k+1}而決定，所以

$$P\left(\underline{\hat{Y}}_{k+2}^{N_c-1} \mid u_{k+1}=j, S_{k+1}=s_{k+1}, \underline{\hat{Y}}_{k+1}, u_k=i, S_k=s_k\right)$$

$$=P\left(\underline{\hat{Y}}_{k+2}^{N_c-1} \mid u_{k+1}=j, S_{k+1}=s_{k+1}\right)$$

$$=\beta_{k+1}^{j}\left(s_{k+1}\right) \qquad (3.24)$$

若再定義

$$\gamma_{j,i}\left(\underline{\hat{Y}}_{k+1}, s_{k+1}, s_k\right)=P\left(u_{k+1}=j, S_{k+1}=s_{k+1}, \underline{\hat{Y}}_{k+1} \mid u_k=i, S_k=s_k\right) \qquad (3.25)$$

（3.23）式可以用後向遞迴的計算方法加以實現：

$$\beta_k^i\left(s_k\right)=\sum_{s_{k+1}} \sum_{j\in\{-1,1\}} \beta_{k+1}^{j}\left(s_{k+1}\right)\gamma_{j,i}\left(\underline{\hat{Y}}_{k+1}, s_{k+1}, s_k\right) \qquad (3.26)$$

如前所述，格子結構的尾巴位元必須迫使編碼器的狀態回到 0，所以設定初始值為

$$\beta_{N_c-1}^{i}\left(s_{N_c-1}\right)=P\left(u_{N_c-1}=i, S_{N_c-1}=s_{N_c-1}, \underline{\hat{Y}}_{N_c-1}\right)$$

$$=\begin{cases}1 & \text{if } s_{N_c-1}=s_b^i(0)\\ 0 & \text{if } s_{N_c-1}=s_b^i(x),\ x\neq 0\end{cases} \qquad (3.27)$$

其中$s_b^i\left(s_k\right)$為在給定了由$S_k=s_k$和$u_{k-1}=i$所定義的路徑之後，所得出s_k的前一個狀態。後向機率遞迴計算的格子架構如圖 3.3 所示。

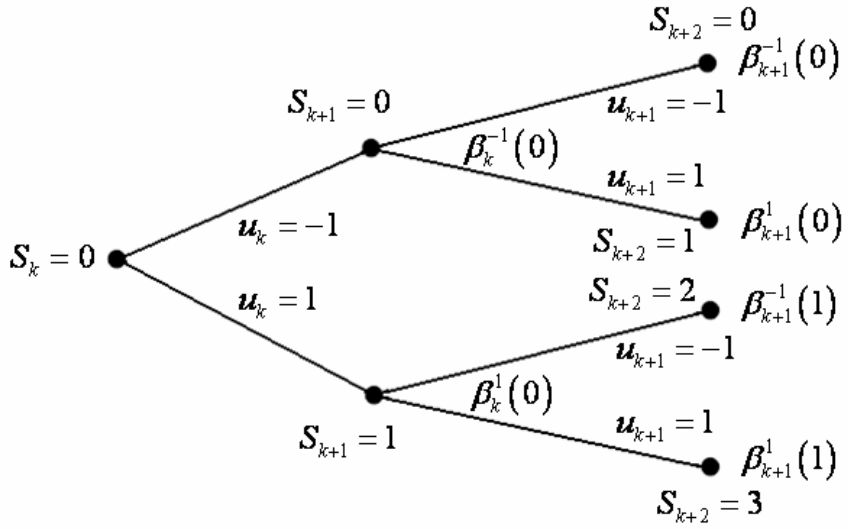

圖 3.3 後向機率計算的格子架構

3.2.3 機率γ的推導

機率$\gamma_{i,j}\left(\underline{\hat{Y}}_k, s_{k-1}, s_k\right)$可以由無記憶性高斯通道的轉移機率（transition probability）和編碼器的格子架構所決定。藉著使用貝氏定理，

$$\begin{aligned}\gamma_{i,j}\left(\underline{\hat{Y}}_k, s_{k-1}, s_k\right) &= P\left(u_k = i, S_k = s_k, \underline{\hat{Y}}_k \mid u_{k-1} = j, S_{k-1} = s_{k-1}\right) \\ &= P\left(\underline{\hat{Y}}_k \mid u_k = i, S_k = s_k, u_{k-1} = j, S_{k-1} = s_{k-1}\right) \\ &\times P\left(u_k = i \mid S_k = s_k, u_{k-1} = j, S_{k-1} = s_{k-1}\right) \\ &\times P\left(S_k = s_k \mid u_{k-1} = j, S_{k-1} = s_{k-1}\right) \qquad (3.28)\end{aligned}$$

對一個無記憶性的高斯通道而言，(3.28) 式的第一項可以寫成：

$$P\left(\hat{\underline{Y}}_k \mid u_k = i, S_k = s_k, u_{k-1} = j, S_{k-1} = s_{k-1}\right)$$

$$= P\left(\hat{\underline{Y}}_k \mid u_k = i, S_k = s_k\right)$$

$$= P\left(\hat{\underline{Y}}_k \mid \underline{Y}_k\right)$$

$$= \prod_{l=0}^{N_l - 1} P\left(\hat{y}_{k,l} \mid y_{k,l}\right)$$

$$= A_k \exp\left[\sum_{l=0}^{N_l - 1} \frac{1}{2} L_c y_{k,l} \hat{y}_{k,l}\right] \qquad (3.29)$$

其中 A_k 在時間 k 為一常數。L_c 為通道狀態消息（channel state information），其定義如下：

$$L_c = 4a \cdot \frac{E_s}{N_0} \qquad (3.30)$$

其中 a 為通道的衰減因子，對於附加性白色高斯雜訊通道（AWGN channel）而言，$a = 1$。

若假設編碼後的位元為獨立相等分佈（independent identically distributed, i.i.d.）在（3.28）式的第二項為：

$$P\left(u_k = i \mid S_k = s_k, u_{k-1} = j, S_{k-1} = s_{k-1}\right) = 0.5 \qquad (3.31)$$

在（3.28）式中的第三項為：

$$P\left(S_k = s_k \mid u_{k-1} = j, S_{k-1} = s_{k-1}\right) = \begin{cases} 1 & \text{if } s_{k-1} = s_b^j\left(s_k\right) \\ 0 & \text{otherwise} \end{cases} \qquad (3.32)$$

綜合以上三項，即結合（3.29）、（3.31）、及（3.32）式，（3.28）式可簡化成為：

$$\gamma_{i,j}\left(\underline{\hat{Y}}_k, s_{k-1}, s_k\right)=\begin{cases} C_k \exp\left[\frac{1}{2}\sum_{l=0}^{N_l-1} L_c y_{k,l}\hat{y}_{k,l}\right] & \text{if } \ s_{k-1}=s_b^j\left(s_k\right) \\ 0 & \text{otherwise} \end{cases} \quad (3.33)$$

其中C_k為一常數。

依此類推，在後向計算機率裡的$\gamma_{j,i}\left(\underline{\hat{Y}}_{k+1}, s_{k+1}, s_k\right)$可以用和$\gamma_{i,j}\left(\underline{\hat{Y}}_k, s_{k-1}, s_k\right)$相似的方法實現，最後可寫成：

$$\gamma_{j,i}\left(\underline{\hat{Y}}_{k+1}, s_{k+1}, s_k\right)=\begin{cases} C'_{k+1} \exp\left[\frac{1}{2}\sum_{l=0}^{N_l-1} L_c y_{k+1,l}\hat{y}_{k+1,l}\right] & \text{if } \ s_{k+1}=s_f^i\left(s_k\right) \\ 0 & \text{otherwise} \end{cases} \quad (3.34)$$

其中C'_{k+1}為一常數，$s_f^i\left(s_k\right)$為給定了由$S_k=s_k$和$u_k=i$所定義的路徑之後，所得出s_k的下一個狀態。

3.2.4 對數相似比率的計算

最後對個別位元u_k而言，其對數相似比率的計算可以總結如以下步驟：

【1】α的初始設定：

$$\alpha_0^i\left(s_0\right)=\begin{cases} P\left(\underline{\hat{Y}}_0 \mid u_0=i, S_0=0\right)P\left(u_0=i\right) & \text{if } s_0=0 \\ 0 & \text{if } s_0\neq 0 \end{cases} \quad (3.35)$$

β的初始設定：

$$\beta_{N_c-1}^i\left(s_{N_c-1}\right)=\begin{cases} 1 & \text{if } s_{N_c-1}=s_b^i\left(0\right) \\ 0 & \text{if } s_{N_c-1}=s_b^i\left(x\right) \ \ x\neq 0 \end{cases} \quad (3.36)$$

【2】前向遞迴機率α的計算：

$$\alpha_k^i(s_k)=\sum_{s_{k-1}}\sum_{j\in\{-1,1\}}\alpha_{k-1}^j(s_{k-1})\gamma_{i,j}\left(\underline{\hat{Y}}_k,s_{k-1},s_k\right) \qquad (3.37)$$

後向遞迴機率β的計算：

$$\beta_k^i(s_k)=\sum_{s_{k+1}}\sum_{j\in\{-1,1\}}\beta_{k+1}^j(s_{k+1})\gamma_{j,i}\left(\underline{\hat{Y}}_{k+1},s_{k+1},s_k\right) \qquad (3.38)$$

【3】u_k的對數相似比率的計算：

$$L_0(u_k)=\log\frac{\sum_{s_k}\alpha_k^1(s_k)\beta_k^1(s_k)}{\sum_{s_k}\alpha_k^{-1}(s_k)\beta_k^{-1}(s_k)} \qquad (3.39)$$

第四章 基於索引的軟性輸出通道解碼

在上一章中，我們主要是以量化輸出索引值之個別位元作為編碼及解碼關注之軸心。在接收端之通道解碼器裡應用基於位元的 BCJR 演算法，得到軟性輸出通道解碼的位元傳送錯誤機率之最佳估測，進而利用第二章的軟性位元解碼，將訊號源量化索引的事前消息一併納入估測計算中。但在通道解碼的過程中，僅是以位元為基礎，利用到的消息也僅至位元的層級，但在索引的層級裡存在比位元層級更多可利用的殘餘冗息。所以在此章中，軟性輸出通道解碼改以索引為解碼之基本單位。接下來，我們將在本章第一節說明以索引為基礎的迴旋編碼器架構，並且指出其與上一章中的迴旋碼編器的相異處，而在第二節，將詳述基於索引的軟性輸出通道解碼演算法的架構。

4.1 基於索引的迴旋編碼器架構

假設在時間 t 時，相對應於訊號源 x_t，一群 M 個的位元組合用以代表索引 $\underline{X}_t \triangleq \{x_{t,0}, x_{t,1}, \ldots, x_{t,M-1}\}$，送進一個由 M_c 個移位暫存器所組成的迴旋編碼器，此基於索引的迴旋編碼器架構圖如圖 4.1 所示。這個迴旋編碼器的暫存器狀態 S_t，類似於上一章所提及的基於位元的迴旋編碼器的暫存器狀態，都是取決於最近 M_c 個輸入值。但不同的是，

上一章所提到的迴旋編碼器是在一個時間單位內輸入一個位元，致使編碼器內的移位暫存器也跟著移動一個位元。而在此章中的迴旋編碼器乃是基於索引編碼為出發點，所以在一個單位時間內，輸入一群足以代表索引的M個位元，致使編碼器裡的移位暫存器在同一單位時間內也移動了M個位元，並且輸出相對應的編碼序列$\underline{Y}_t \triangleq \{y_{t,0}, y_{t,1}, \ldots, y_{t,MN_l-1}\}$，其中$M$和$MN_l$的比值$\left(r = \frac{M}{MN_l} = \frac{1}{N_l}\right)$為通道編碼率。值得注意的是，有一個條件關係必須滿足，即$M_c \geq M$。否則，編碼器裡的暫存器數量連代表一個索引的位元數量都不夠，便無法在一個單位時間內，針對一個索引作通道編碼的動作，也就失去了基於索引編碼的主要精神。還有在此的移位暫存器個數M_c也未必一定要為索引位元組合個數M的整數倍，換句話說，就是編碼器裡的暫存器所儲存的可以不全都是完整的索引，也可能摻雜了一索引裡零碎的位元組合，而且其暫存器的狀態值由每個單位時間內輸入的索引位元組合完全進入暫存器後始可得。

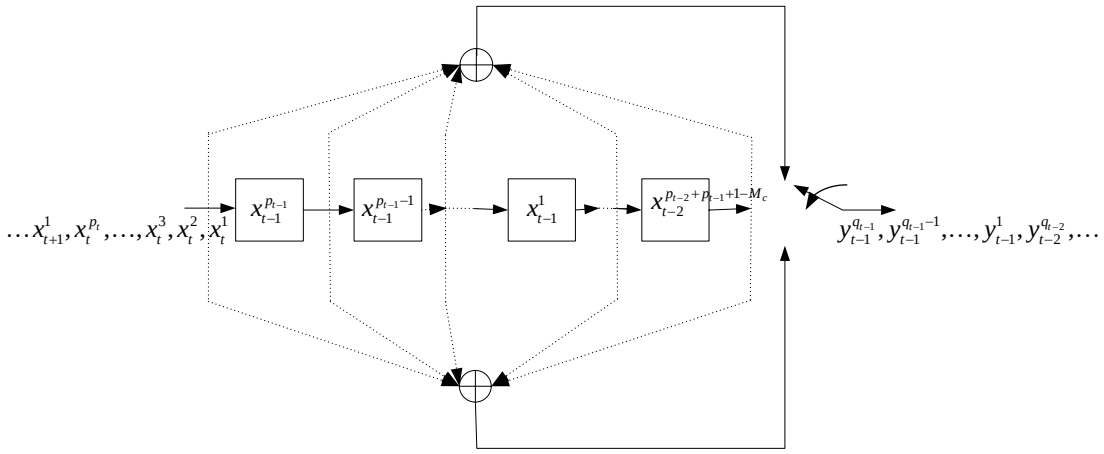

圖 4.1 基於索引的迴旋編碼器架構

4.2 基於索引的 BCJR 演算法

依照上一節中基於索引的通道編碼器的架構，假設以 N_s 個索引為格子架構裡一個主要的消息區塊，並且經由通道編碼器傳送至可加性白色高斯雜訊通道（AWGN channel）中。在接收端，通道解碼器將收到實數序列 $\hat{\underline{Y}}_0^{N_s-1} \triangleq \left\{\hat{\underline{Y}}_0, \hat{\underline{Y}}_1, \ldots, \hat{\underline{Y}}_{N_s-1}\right\}$，其中 $\hat{\underline{Y}}_t \triangleq \left\{\hat{y}_{t,0}, \hat{y}_{t,1}, \ldots, \hat{y}_{t,MN_l-1}\right\}$，而且

$$\hat{y}_{t,l} = y_{t,l} + n_{t,l},\ l = 0,1,\ldots MN_l - 1 \qquad (4.1)$$

其中 $n_{t,l}$ 為零平均值，且變異數為 σ^2 的高斯雜訊取樣。

每一個解碼後索引 x_t 的事後機率為

$$P\left(x_t = i \mid \hat{\underline{Y}}_0^{N_s-1}\right) = \sum_{S_t} \lambda_t^i\left(s_t\right) \qquad (4.2)$$

其中 $\lambda_t^i\left(s_t\right) = P\left(x_t = i, S_t = s_t \mid \hat{\underline{Y}}_0^{N_s-1}\right),\ i = 0,1,\ldots,2^M - 1$。藉由貝氏定理，可得：

$$\lambda_t^i(s_t)=\frac{P\left(x_t=i,S_t=s_t,\hat{\underline{Y}}_0^{N_s-1}\right)}{P\left(\hat{\underline{Y}}_0^{N_s-1}\right)}$$

$$=\frac{P\left(x_t=i,S_t=s_t,\hat{\underline{Y}}_0^{N_s-1}\right)}{\sum_i\sum_{s_t}P\left(x_t=i,S_t=s_t,\hat{\underline{Y}}_0^{N_s-1}\right)}$$

$$=\frac{P\left(x_t=i,S_t=s_t,\hat{\underline{Y}}_0^{t},\hat{\underline{Y}}_t^{N_s-1}\right)}{\sum_i\sum_{s_t}P\left(x_t=i,S_t=s_t,\hat{\underline{Y}}_0^{N_s-1}\right)}$$

$$=\frac{P\left(x_t=i,S_t=s_t,\hat{\underline{Y}}_0^{t}\right)P\left(\hat{\underline{Y}}_t^{N_s-1}\mid x_t=i,S_t=s_t,\hat{\underline{Y}}_0^{t}\right)}{\sum_i\sum_{s_t}P\left(x_t=i,S_t=s_t,\hat{\underline{Y}}_0^{N_s-1}\right)}$$

$$=\frac{\alpha_t^i(s_t)\beta_t^i(s_t)}{\sum_i\sum_{s_t}\alpha_t^i(s_t)\beta_t^i(s_t)} \qquad (4.3)$$

其中α和β的定義為

$$\alpha_t^i(s_t)=P\left(x_t=i,S_t=s_t,\hat{\underline{Y}}_0^{t}\right) \qquad (4.4)$$

$$\beta_t^i(s_t)=P\left(\hat{\underline{Y}}_t^{N_s-1}\mid x_t=i,S_t=s_t\right) \qquad (4.5)$$

類似於第三章中基於位元的軟性輸出通道解碼演算法裡的α和β，在此，基於索引的前向機率α和後向機率β亦可以遞迴計算的形式呈現：

$$\alpha_t^i(s_t)=\sum_{s_{t-1}}\sum_j\alpha_{t-1}^j(s_{t-1})\gamma_{i,j}\left(\hat{\underline{Y}}_t,s_{t-1},s_t\right) \qquad (4.6)$$

$$\beta_t^i(s_t)=\sum_{s_{t+1}}\sum_j\beta_{t+1}^j(s_{t+1})\gamma_{j,i}\left(\hat{\underline{Y}}_{t+1},s_{t+1},s_t\right) \qquad (4.7)$$

α和β的初始條件將於下一小節中詳加說明，而其中$\gamma_{i,j}$可被展開表

示為：

$$\gamma_{i,j}\left(\underline{\hat{Y}}_t, s_t, s_{t-1}\right) = P\left(x_t = i, S_t = s_t, \underline{\hat{Y}}_t \mid x_{t-1} = j, S_{t-1} = s_{t-1}\right)$$

$$= P\left(\underline{\hat{Y}}_t \mid x_t = i, S_t = s_t, x_{t-1} = j, S_{t-1} = s_{t-1}\right)$$

$$\times P\left(x_t = i \mid S_t = s_t, x_{t-1} = j, S_{t-1} = s_{t-1}\right)$$

$$\times P\left(S_t = s_t \mid x_{t-1} = j, S_{t-1} = s_{t-1}\right) \qquad (4.8)$$

對於一個無記憶性的通道（memoryless channel），在（4.8）式中的第一項可被寫成：

$$P\left(\underline{\hat{Y}}_t \mid x_t = i, S_t = s_t, x_{t-1} = j, S_{t-1} = s_{t-1}\right)$$

$$= P\left(\underline{\hat{Y}}_t \mid x_t = i, S_t = s_t\right)$$

$$= P\left(\underline{\hat{Y}}_t \mid x\left(s_t, i\right)\right)$$

$$= K_t \exp\left[\frac{2}{\sigma^2}\sum_{l=0}^{MN_l-1} y_{t,l}\cdot\hat{y}_{t,l}\right] \qquad (4.9)$$

其中 K_t 為一常數，而 $x\left(s_t, i\right)$ 為代表了在給定狀態 s_t 和輸入索引值 i 之後的輸出索引。接著，考慮（4.8）式中的第二項機率，其意謂著訊號源量化後之前後相鄰兩索引間的關聯性，大致可分為如下列的情況討論之：

1） 若量化後之相鄰索引間，沒有存在殘餘冗息，亦即被傳送的索引彼此之間為獨立相等分佈（i.i.d.），因此（4.8）式中的第二項機率可寫為：

$$P\left(x_t = i \mid S_t = s_t, x_{t-1} = j, S_{t-1} = s_{t-1}\right)$$

$$= P\left(x_t = i\right) = \frac{1}{2^M} \qquad (4.10)$$

2） 當索引間的殘餘冗息被納入考量時，也就是將利用訊號源經過量化後，因為量化器無法完全地去除量化後索引間所存在的相關性，因此可得

$$P\left(x_t = i \mid S_t = s_t, x_{t-1} = j, S_{t-1} = s_{t-1}\right)$$

$$= P\left(x_t = i \mid x_{t-1} = j\right) \qquad (4.11)$$

而此機率值，可經由事前經大量語料訓練而得。

在（4.8）式中的第三項機率為：

$$P\left(S_t = s_t \mid x_{t-1} = j, S_{t-1} = s_{t-1}\right) = \begin{cases} 1 & \text{if } s_{t-1} = s_b^j\left(s_t\right) \\ 0 & \text{otherwise} \end{cases} \qquad (4.12)$$

其中 $s_b^j\left(s_t\right)$ 為在給定了由 $S_t = s_t$ 和 $x_{t-1} = j$ 所定義的路徑之後，所得出 s_t 的前一個狀態。此項機率決定了格子結構中，時間點與相鄰時間點之間狀態的轉移和路徑的行進方向。在（4.7）式中的 $\gamma_{j,i}\left(\underline{\hat{Y}}_{t+1}, s_{t+1}, s_t\right)$ 如同（4.6）中的 $\gamma_{i,j}\left(\underline{\hat{Y}}_t, s_t, s_{t-1}\right)$ 可依照類似於如上所描述的方法而求得。

在求得了每一個解碼後索引 x_t 的事後機率 $P\left(x_t = i \mid \underline{\hat{Y}}_0^{N_s-1}\right)$ 之後，接著，我們將所有可能的解碼後索引 $x_t = i$，$i = 0,1,\ldots,2^M - 1$，的事後機率 $P\left(x_t = i \mid \underline{\hat{Y}}_0^{N_s-1}\right)$ 代入最小均方錯誤準則，估測出重建後的參數 $\hat{v}_t$ 如下所示：

$$\hat{v}_t = \sum_{x_t \in J} c_{x_t} \cdot P\left(x_t = j \mid \underline{\hat{Y}}_0^{N_s-1}\right),\ J \in \left\{0,1,\ldots.2^M - 1\right\} \qquad (4.13)$$

其中的c_{x_t}為索引x_t所對應的量化參數值。

4.3 前向和後向機率的初始條件設定

這一節中，將詳細說明以索引為基礎的軟性輸出解碼演算法在初始值的設定。而由於此一系統是架構於迴旋編碼的基礎上，所以初始值的設定主要是源自於迴旋編碼過程中格子架構路徑圖所存在的限制，包括起始狀態為零或填入額外的訊號迫使結束狀態回歸到零的相關條件。把每個限制合理地融入演算法內，拒絕不可能的發生的路徑與狀態，是維持解碼程序之可靠性很重要的環節。

前向機率之初始定義為$\alpha_0^i(s_0)=P\left(x_0=i,S_0=s_0,\underline{\hat{Y}}_0\right)$，表示初始狀態$S_0=s_0$、傳輸索引為$i$與輸出訊號為$\underline{\hat{Y}}_0$的結合機率，透過條件機率展開以及初始狀態為零的限制可得其設定為

$$\begin{aligned}\alpha_0^i(s_0)&=P\left(\underline{\hat{Y}}_0\mid x_0=i,S_0=s_0\right)P\left(x_0=i\mid S_0=s_0\right)P\left(S_0=s_0\right)\\&=\begin{cases}P\left(\underline{\hat{Y}}_0\mid x_0=i,S_0=0\right)P\left(x_0=i\right) & \text{if } s_0=0\\0 & \text{if } s_0\neq 0\end{cases}\end{aligned}\qquad(4.14)$$

其中已排除了初始狀態不為零的事件。

欲得後向機率的初始$\beta_{N_s-1}^i\left(s_{N_s-1}\right)$的設定，先檢視在時刻$t=N_s-1$時所要計算的事後機率$\lambda_{N_s-1}^i\left(s_{N_s-1}\right)=P\left(x_{N_s-1}=i,S_{N_s-1}=s_{N_s-1}\mid\underline{\hat{Y}}_0^{N_s-1}\right)$以

及前向機率 $\alpha^{i}_{N_s-1}\left(s_{N_s-1}\right)=P\left(x_{N_s-1}=i,S_{N_s-1}=s_{N_s-1},\hat{\underline{Y}}_0^{N_s-1}\right)$，利用貝氏定理將事後機率 $\lambda^{i}_{N_s-1}\left(s_{N_s-1}\right)$ 展開後，結果可以只用 $\alpha^{i}_{N_s-1}\left(s_{N_s-1}\right)$ 完整表示，進一步再與上一節計算事後機率之定義（4.3）相比較，整理如下

$$\begin{aligned}\lambda^{i}_{N_s-1}\left(s_{N_s-1}\right)&\triangleq P\left(x_{N_s-1}=i,S_{N_s-1}=s_{N_s-1}\,|\,\hat{\underline{Y}}_0^{N_s-1}\right)\\&=\frac{P\left(x_{N_s-1}=i,_{N_s-1}=s_{N_s-1},\hat{\underline{Y}}_0^{N_s-1}\right)}{\sum_i\sum_{s_{N_s-1}}P\left(x_{N_s-1}=i,_{N_s-1}=s_{N_s-1},\hat{\underline{Y}}_0^{N_s-1}\right)}\\&=\frac{\alpha^{i}_{N_s-1}\left(s_{N_s-1}\right)}{\sum_i\sum_{s_{N_s-1}}\alpha^{i}_{N_s-1}\left(s_{N_s-1}\right)}\\&\triangleq\frac{\alpha^{i}_{N_s-1}\left(s_{N_s-1}\right)\beta^{i}_{N_s-1}\left(s_{N_s-1}\right)}{\sum_i\sum_{s_{N_s-1}}\alpha^{i}_{N_s-1}\left(s_{N_s-1}\right)\beta^{i}_{N_s-1}\left(s_{N_s-1}\right)}\end{aligned}\qquad(4.15)$$

故滿足上式機率運算的合理設定為 $\beta^{i}_{N_s-1}\left(s_{N_s-1}\right)=1$。再探討於時刻 $t=N_s-2$ 時後向機率 $\beta^{i}_{N_s-2}\left(s_{N_s-2}\right)$ 的計算，先檢視其機率之定義 $\beta^{i}_{N_s-2}\left(s_{N_s-2}\right)=P\left(\hat{\underline{Y}}_{N_s-1}^{N_s-1}\,|\,x_{N_s-2}=i,S_{N_s-2}=s_{N_s-2}\right)$ 與後向遞迴式中所定義之過渡機率 $\gamma_{j,i}\left(\hat{\underline{Y}}_{N_s-1},s_{N_s-1},s_{N_s-2}\right)$，利用邊緣機率將 $\beta^{i}_{N_s-2}\left(s_{N_s-2}\right)$ 展開後，結果可單以過渡機率完整表示，進一步與上一節計算後向機率之遞迴關係式（4.7）相比較，其整理如下

$$
\begin{aligned}
&\beta^{i}_{N_s-2}\left(s_{N_s-2}\right)\\
&=\sum_{j}\sum_{s_{N_s-1}} P\left(x_{N_s-1}=j, S_{N_s-1}=s_{N_s-1}, \underline{\hat{Y}}_{N_s-1} \mid x_{N_s-2}=i, S_{N_s-2}=s_{N_s-2}\right)\\
&=\sum_{j}\sum_{s_{N_s-1}} \gamma_{j,i}\left(\underline{\hat{Y}}_{N_s-1}, s_{N_s-1}, s_{N_s-2}\right)\\
&\triangleq\sum_{j}\sum_{s_{N_s-1}} \beta^{j}_{N_s-1}\left(s_{N_s-1}\right)\times\gamma_{j,i}\left(\underline{\hat{Y}}_{N_s-1}, s_{N_s-1}, s_{N_s-2}\right) \qquad (4.16)
\end{aligned}
$$

此結果亦顯示 $\beta^{i}_{N_s-1}\left(s_{N_s-1}\right)=1$ 之設定是符合遞迴關係式之機率運算。此外，過渡機率 $\gamma_{j,i}\left(\underline{\hat{Y}}_{N_s-1}, s_{N_s-1}, s_{N_s-2}\right)$ 的計算應配合迴旋編碼需額外傳送歸零訊號的條件，利用設定訊號的發生的機率以拒絕不可發生的解碼路徑，過渡機率的展開式整理如下

$$
\begin{aligned}
&\gamma_{j,i}\left(\underline{\hat{Y}}_{N_s-1}, s_{N_s-1}, s_{N_s-2}\right)\\
&=P\left(x_{N_s-1}=j, S_{N_s-1}=s_{N_s-1}, \underline{\hat{Y}}_{N_s-1} \mid x_{N_s-2}=i, S_{N_s-2}=s_{N_s-2}\right)\\
&=P\left(\underline{\hat{Y}}_{N_s-1} \mid x_{N_s-1}=j, S_{N_s-1}=s_{N_s-1}, x_{N_s-2}=i, S_{N_s-2}=s_{N_s-2}\right)\\
&\times P\left(x_{N_s-1}=j \mid x_{N_s-2}=i, S_{N_s-1}=s_{N_s-1}, S_{N_s-2}=s_{N_s-2}\right)\\
&\times P\left(S_{N_s-1}=s_{N_s-1} \mid S_{N_s-2}=s_{N_s-2}, x_{N_s-2}=i\right) \qquad (4.17)
\end{aligned}
$$

其中傳輸訊號 $x_{N_s-1}=0$ 屬於額外加入的歸零訊號，是必然的事件，所以 $x_{N_s-1}\neq 0$ 的發生機率將設定為零。同理，在其他時刻都應考量是否為歸零訊號以給予同樣的設定。

由於解碼的遞迴演算法建構在迴旋編碼先天基礎上，須在演算過程之初始給予相對應的條件設定，並藉由機率運算驗證設定之合理性。在系統模擬的實驗中，證實了初始的條件設定對於解碼器能否獲

得穩定的結果有極大的影響。

第五章 實驗模擬與結果分析

在前面的章節我們描述了基於位元和基於索引的兩種軟性位元解碼演算法之理論推導與系統架構。在本章中，進一步將演算法應用於系統模擬，以期藉由系統之模擬結果驗證演算法之正確性。

本章的內容將分為三小節來進行實驗之模擬與結果之分析討論。第一節模擬基於位元的軟性輸出通道解碼之系統架構，同時探討運用不同程度之量化索引殘餘冗息，於解碼端效能評比與分析。在第二節中，將模擬基於索引的軟性輸出通道解碼之通訊系統，同樣地，考量使用不同程度的量化索引殘餘冗息進行探討與效能分析。最後，在第三節中，將討論前兩節中不同架構之系統在整體結果的分析與比較，以提供通訊系統設計之參考。

5.1 基於位元的軟性輸出通道解碼系統之模擬

5.1.1 系統模擬之步驟說明

在此節之內容中，主要將結合第二及第三章之演算法進行系統模擬與結果分析。系統中訊號源v_t取自一階自迴歸處理（first-order autoregressive process），且其訊號源之相關因子（correlation factor）採用$\rho_{vv}=0.9$，訊號源之變異數則設定為$\sigma_v^2=1$。關係式表

示為：

$$v_t = \rho_{vv} v_{t-1} + n_t \qquad (5.1)$$

其中n_t表均值為零的白色高斯雜訊取樣。在每次的模擬過程中，由訊號源產生出10000點的資料作為系統之輸入，接著將產生出之訊號源送進量化器中，所採用之量化器為純量 Lloyd-Max 量化器（scalar LMQ）[9]，而且量化索引使用$M=2$位元。由於研究重點不在於量化索引的最佳位元映對方法（bit mapping scheme），所以選用自然二位元編碼方式（natural binary code, NBC）[10]。在得到量化索引的位元資料後，將其通過二位元相位鍵移（BPSK）調變，把每一位元由原來的$\{0,1\}$映對成$\{1,-1\}$。下一步，以200個消息位元與額外附加迫使狀態回歸至零的冗餘位元作為一個格子架構之區塊長度，依序送進迴旋通道編碼器中，其中4個冗餘位元的擇定是配合系統中迴旋編碼器採用通道編碼率為$r=1/2$和限制長度為$L=5$的架構。接著，便將編碼後的 BPSK 訊號經由通道傳送至接收端，模擬的通道環境使用白色高斯雜訊通道（AWGN channel），即在傳送的過程中，僅加入白色高斯雜訊以作通訊干擾。

在接收端，接收器所接收到的不再是當初傳送時的位元$\{1,-1\}$，而是經由高斯雜訊擾亂後的實數值，將其輸入到基於位元的軟性輸出通道解碼器，解碼後將產生我們想要的硬性判定和其可靠性，此處必

須注意在軟性輸出通道解碼的過程中，須以格子構架之區塊長度為演算法之實行間隔區塊，每一區塊所解出之軟性位元訊息，將對等於傳送時所包含的消息位元及冗餘位元的訊息，而其中所要的僅有消息位元之訊息，所以必須將多餘的冗餘位元之訊息捨棄。有了通道解碼器產生出的軟性位元訊息後，便可利用軟性位元解碼演算法進行估測和重建參數。

作軟性位元解碼演算法估測時，因為實驗中，使用了不同階級的事前消息而有下列三種試驗情況，實驗之結果如圖 5.1 所示：

1） 硬性判定（HD）：

為最簡單之估測判定，直接利用軟性輸出通道解碼器所提供硬性判定之可靠度，求出可能傳送位元的對數相似比率：

$$L_0\left(x_{t,m}\right)=\log\frac{P\left(x_{t,m}=1\middle|\hat{\underline{Y}}_0^{N_c-1}\right)}{P\left(x_{t,m}=-1\middle|\hat{\underline{Y}}_0^{N_c-1}\right)} \qquad (5.2)$$

然後直接利用此對數相似比率對可能傳送之位元 $x_{t,m}$ 作硬性判定，得出：

$$\tilde{x}_{t,m}=\operatorname{sign}\left[L_0\left(x_{t,m}\right)\right] \qquad (5.3)$$

得到每個可能傳送之位元的硬性判定後，再將所得的 M 位元組合視為一索引作解量化（dequantization）之動作，即反查量化表，估測重建的參數 $\hat{v}_t$。

2） 零階事前消息（AK0）的計算方法：

在收到軟性輸出通道解碼所得的軟性位元訊息後，加入事先已訓練的零階事前消息，比照（2.13）式作軟性位元解碼估測，可得重建參數$\hat{v}_t$。

3） 一階事前消息（AK1）的計算方法：

如同零階事前消息（AK0）的計算方法，加入事先已訓練的一階事前消息，利用（2.14）式進行軟性位元解碼估測，得重建參數$\hat{v}_t$。

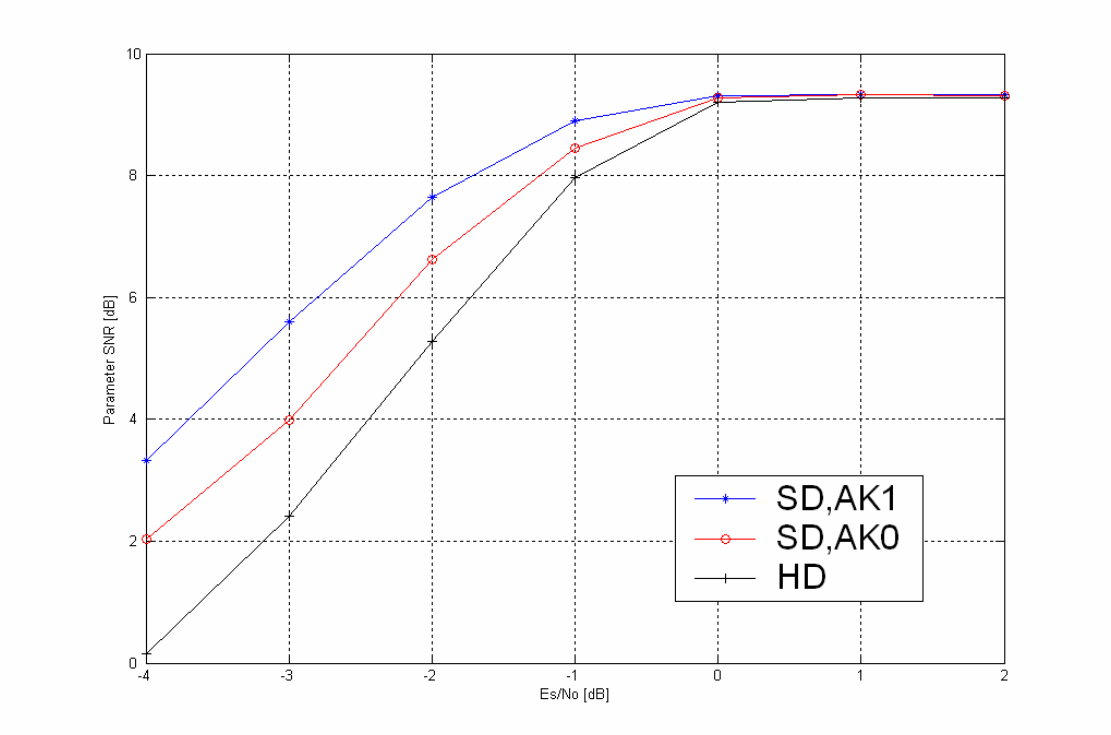

圖 5.1 基於位元的軟性位元解碼之系統模擬

5.1.2 結果分析

圖 5.1 中，系統之評估為使用訊號源參數v_t和重建參數$\hat{v}_t$之間的訊號雜訊比（signal-to-noise ratio, SNR）作為衡量：

$$\mathrm{SNR} = 10\log_{10}\frac{v_t^2}{\left(v_t - \hat{v}_t\right)^2} \quad [\mathrm{dB}] \qquad (5.4)$$

在圖中，可見當通道情況較差時，即訊號能量對通道雜訊能量比率較低時，硬性判定很明顯地衰退最快。如果我們在軟性位元解碼時加上零階事前消息進行解碼，可得知在通道較差時，衰退的程度就沒有硬性判定那麼大，系統的效能明顯地被提升上來。若利用了更多訊息源索引間的殘餘冗息，如一階事前消息，由實驗結果得知，通道較差的情況，其效能比使用零階事前消息更好，系統的效能再次地被往上提升。順便一提，在圖中當訊號能量對通道雜訊能量比率較高時，即通道的情況被雜訊干擾不嚴重時，三者的效能差不多。即效能值被約束在一定的值，此值為量化器所使用的量化位元數$M = 2$所影響，本實驗中此臨界值為9.351464 dB 。

5.2 基於索引的軟性輸出通道解碼系統之模擬

5.2.1 系統模擬之步驟說明

基於索引之系統模擬，主要是針對第四章所提之演算法進行實驗和模擬，類似於上一節所描述的基於位元之系統模擬，以下將指出其

同異之處。在訊號源的產生與基於位元之系統一樣，取自於一階自迴歸訊號源，資料量為10000點，同樣送進純量 Lloyd-Max 量化器，每一點量化後產生出2位元的索引，接著作二位元相位鍵移調變，再送入通道編碼器。在此需注意的是，在送入通道編碼器之前，所使用的模擬平台皆與上一節所描述完全相同，但在此節中，乃基於索引為主要編碼解編之基本單元，所以在上一節中使用200個消息位元加上4個冗餘位元為格子架構之區塊長度，在此節中必須更改為100個消息索引加上2個冗餘索引為格子架構之區塊長度，雖然迴旋編碼器之架構與上一節所提及沒有不同，但所關注及切入之焦點不同，這將會影響之後接收端所做之通道解碼動作。

通道編碼後之位元組合在通過可加性白色高斯雜訊通道之後，在接收端，相對於傳送端之編碼器設計，通道解碼器利用軟性輸出通道解碼的演算法架構，必須遵循著以每一個格子架構區塊長度為實行間隔區塊，此處每一個格子架構區塊解出100個消息索引和2個冗餘索引的軟性輸出訊息。而實際上，在通道解碼器的輸出端會直接移除額外加入的冗餘索引之軟性輸出訊息，而僅留下有用的100個消息索引的軟性輸出訊息，以送進最小均方錯誤估測器進行重建估測。

在做最小均方錯誤估測時，又因為使用了不同份量的訊號源量化後索引的冗餘訊息，即不同階級的事前消息，所以實驗模擬又分成下

列兩種情況：

1） 無事前消息（NAK）的方法：

假設量化後索引間，無存在殘餘冗息，即彼此獨立相等分佈，可得（4.10）式，將其代入前向和後向機率的計算中，可求出索引 x_t 的事後機率，進一步利用最小均方錯誤準則估測重建參數 $\hat{v}_t$。

2） 一階事前消息（AK1）的方法：

利用100000點量化後索引訓練出一階事前消息，即為(4.11)式，同樣地將此機率利用到前向和後向機率之計算，如此求得蘊含了量化後索引間之冗餘訊息的解碼後索引 x_t 的事後機率，再代入最小均方錯誤準則，得估測後之重建參數 $\hat{v}_t$。

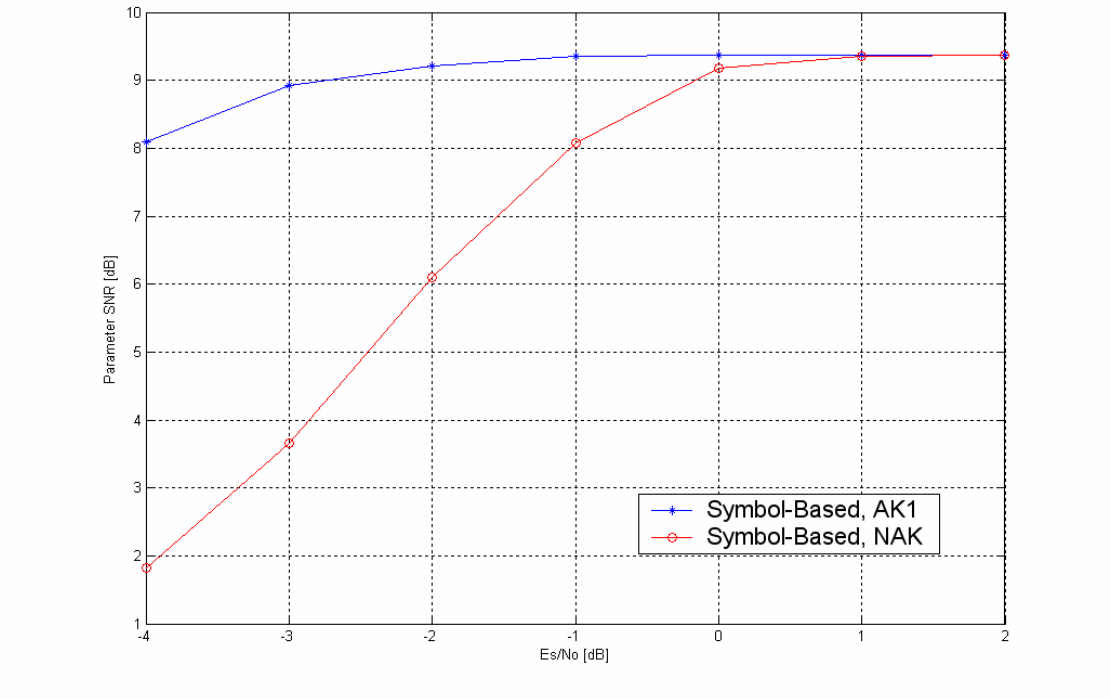

圖 5.2 基於索引的軟性位元解碼之系統模擬

5.2.2 結果分析

圖 5.2 之系統評估，同樣使用（5.4）式作為衡量，由圖中結果在通道環境較差時，添加使用了一階事前消息（AK1）的系統比使用無事前消息（NAK）的系統的效能還要好，即說明了利用到量化後索引間之殘餘冗息的估測系統較能有效抵抗通道雜訊對系統之干擾，而當訊號能量對通道雜訊能量比在一定程度以上時，兩種方法的系統效能會顯得差不多，而且其最佳效能將被量化器之量化位元數量 $M=2$ 所拘限，對於此系統而言，同於上一節基於位元之系統模擬，其臨界值為9.351464 dB。

5.3 基於位元與基於索引之系統綜合結果分析和比較

在前兩節中，系統的架構主要是圍繞在不同的處理單元，進行編碼、傳送、解碼之動作。系統的架構基礎由 5.1 節的位元轉換到 5.2 節的索引，乃是因為我們期望能在索引為層級的架構下，獲得比位元為層級的架構更多的殘餘冗息，進而利用這些消息量，改善系統之效能，讓系統對通道之雜訊干擾更具抵抗性。因此，在這一節將比較 5.1 節與 5.2 兩者之間的效能分析，並且證實以索引為處理層級之架構確實讓通訊系統有更好的效能，並且對於通道雜訊更具有抵抗性。

首先，由圖 5.1 之結果，可看出基於位元且使用 AK0 方法的軟性

位元解碼的系統的整體效能，比使用硬性判定方法的系統效能還要來得好，可得知使用了量化後索引間之殘餘冗息的系統比完全不使用殘餘冗息之系統其效能更加優益。再者比較圖 5.1 中使用 AK0 方法的軟性位元解碼以及圖 5.2 中 NAK 方法的軟性位元解碼之兩結果，將其繪於圖 5.3，可看出兩方法之效能相差不遠，而基於索引之系統僅使用無事前消息的殘餘冗息，但基於位元之系統已使用到了零階事前消息的殘餘冗息，所以由此可知基於索引之系統在使用殘餘冗息時提升效能的速度較有效率。

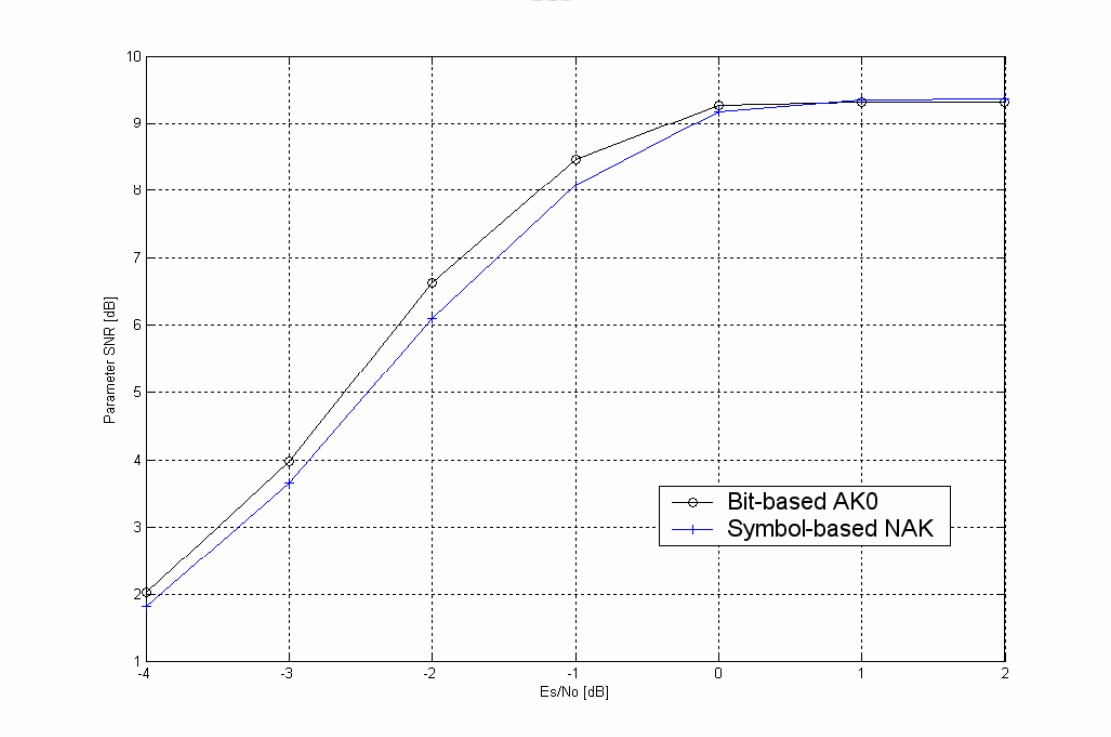

圖 5.3 基於位元與索引系統之比較

最後，比較圖 5.1 中使用 AK1 方法的軟性位元解碼系統，和圖

5.2 中使用 AK1 方法的軟性位元解碼系統，將其兩系統之效能繪於圖 5.4 中，結果明顯的知道以索引為基礎的架構效能遠高於以位元為型礎的架構。所以，由以上之效能綜合分析比較，我們的確可證實出以索引為層級之架構確實讓通訊系統有更好的效能和對於通道雜訊更具有抵抗性。

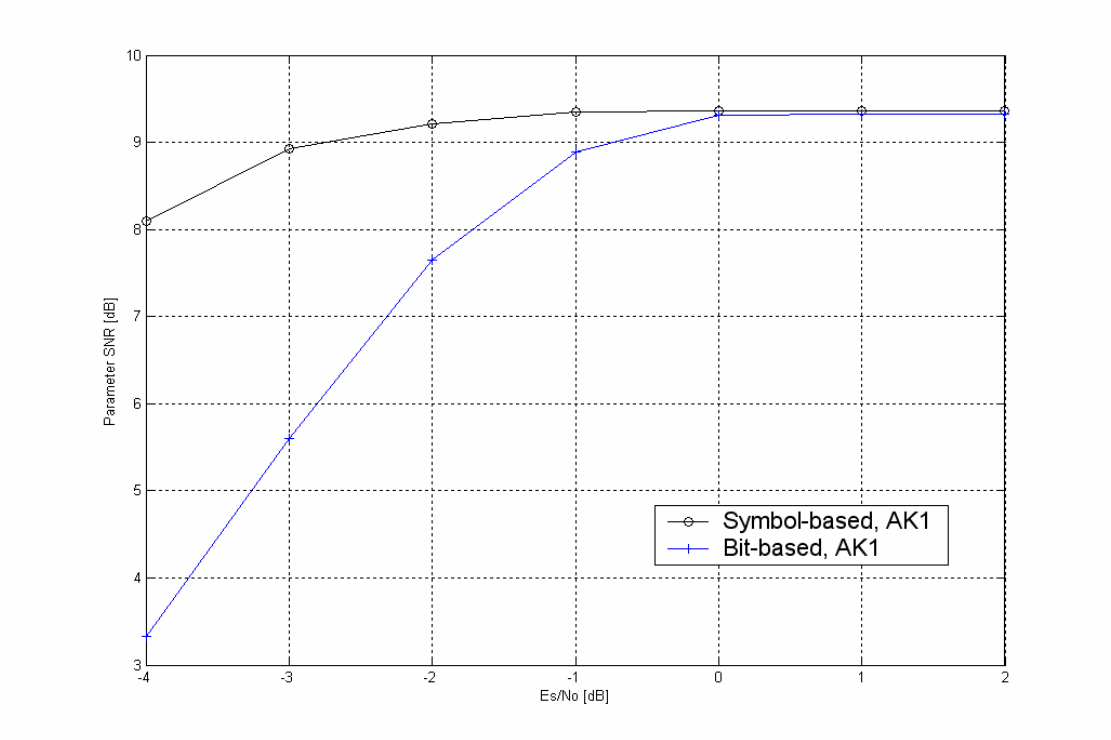

圖 5.4 結合 AK1 與不同解碼之比較

第六章 結論與未來展望

6.1 結論

本論文主要探討軟性位元解碼在傳輸於雜訊通道下之錯誤隱匿，其肇因於數位通訊系統傳輸過程中，經歷了傳輸通道裡不可抗拒的雜訊干擾，致使在接收端經過通道解碼後，仍舊殘餘了位元錯誤。因此，由傳統接收端的硬性判定，改以添加入關於此硬性判定的可靠度，而轉以稱為軟性位元解碼的方式在接收端進行判定與估測之工作。此演算法挖掘並利用了量化後索引間的殘餘冗息，而且由於在執行軟性位元解碼演算法時，需要關於硬性判定的可靠消息。因而在此使用了提供最佳事後機率的 BCJR 演算法，來供給我們需要的位元可靠消息。但傳統的 BCJR 演算法乃以位元為解碼之基本架構，為了得到更好的系統效能與抵抗通道雜訊的能力，因此我們提出一種利用混合位元與索引層級的相關性的改良型 BCJR 演算法。最後，在實驗中，我們使用馬可夫程序為訊號源，並且由實驗模擬可証實改良後的 BCJR 演算法結合量化後索引的一階事前消息確實可得最佳的錯誤隱匿效能。

6.2 未來展望

在此論文中，通道之模擬僅以基本的可加性白色高斯雜訊通道為模擬的對象，說明所提之演算法確實可獲得改善效能的助益，但面對真實多變的現實環境，若僅是以可加性白色高斯雜訊作為通道的模擬，並非足以應付。因此在未來，可以針對通道之模擬加以改善，並且善加利用通道之特性，結合在此所提之演算法，以盼能得到更佳之錯誤隱匿的效果，以增進人們生活之便利。

參考文獻

[1] C.E. Shannon, “A Mathematical Theory of Communication,” *Bell Sust. Tech. J.*, vol. 27, pp. 379-423, pp. 623-656, 1948.

[2] K. Sayood and J. C. Borkenhagen, “Use of residual redundancy in the design of joint source/channel coders,” *IEEE Trans. Commun.*, vol. 39, No. 6, pp. 838-846, June 1991.

[3] T. Fingscheidt and P. Vary, “Robust speech coding: A universal approach to bit error concealment,” in *Proc. IEEE ICASSP*, vol. 3, pp. 1667-1670, Apr. 1997.

[4] T. Fingscheidt and P. Vary, “Softbit Speech Decoding: A New Approach to Error Concealment,” *IEEE Trans. Speech Audio Processing*, vol. 9, No. 3, Mar. 2001.

[5] J. Hagenauer, “Source-Controlled Channel Decoding,” *IEEE Trans. Commun.*, vol. 43, pp. 2449-2457, Sept. 1995.

[6] L. R. Bahl, J. Cocke, F. jelinek and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inform. Theory*, vol. 20, pp. 284-287, Mar. 1974.

[7] J. Hagenauer and P. Hoeher, “A viterbi algorithm with soft-decision output and its application,” in *Proc. IEEE Global Telecommunications Conf. (GLOBECOM)*, vol. 3, pp. 1680-1686, Nov. 1989.

[8] S. S. Pietrobon and S. A. Barbulescu, “A simplification of the modified Bahl decoding algorithm for systematic convolutional codes,” in *Proc. Int. Symp. on Inform. Theory and its Applications*, pp. 1073-1077, Nov. 1994. Revised 4 Jan. 1996.

[9] J. Max, “Quantizing for minimum distortion,” *IRE Trans. Inform. Theory*, vol. IT-6, pp. 7-12, Mar. 1960.

[10] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*. Englewood

Cliffs, NJ: Prentice-Hall, 1984.