

國立交通大學

電信工程學系

碩士論文



符號冗息在訊源控制通道解碼之研究
A study of symbol-level source-controlled channel
decoding

研究生：傅泰魁

指導教授：張文輝博士

中華民國九十五年六月


符號冗息在訊源控制通道解碼之研究

學生：傅泰魁

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



在數位傳輸系統的設計中，合併訊源通道解碼的研究旨在補償通道錯誤所造成的解碼失真。訊號通過訊源編碼器之後，在量化輸出位元串之間存在某種形式的關聯性，若能在通道解碼器妥善地利用這一項事前資訊，將可對接收端的解碼能力有所提升。我們提出了一個訊源控制通道解碼機制，在接收端將利用量化後所殘留的訊源冗息提昇通道解碼工作的正確性。在這架構中，有別於以往以單一位元為基礎的演算法，我們探討以符號為層級的改良式 BCJR 演算法，計算每個符號的後驗機率後再做最大後驗機率的估測。有關論文的實驗模擬，先以自迴歸訊號源做驗證，再將此架構應用於現今被廣泛地使用的分散式語音辨識系統上作討論分析。

A study of symbol-level source-controlled channel decoding

Student: Tai-Kuei Fu Advisor: Dr. Wen-Whei Chang

Department of Communication Engineering

National Chiao Tung University

Abstract

For digital multimedia communication, the system performance will degrade after compressed signals are transmitted over noisy channels. In order to combat channel errors, the signals will be protected by means of joint source-channel coding methods. When signals are quantized by the source encoder into bit sequences, there exists residual redundancy, due to non-uniform distribution and correlations between them. By taking advantage of this a priori knowledge, we propose a source-controlled channel decoding scheme which incorporates the residual redundancy into the channel decoding process. Also proposed is a modified BCJR algorithm that computes the a posteriori probability at symbol level, as opposed to conventional bit-level BCJR algorithm. Experiments were carried out to investigate the robustness of the proposed coding approach. The input signals considered here include first-order autoregressive sources and speech parameters used for distributed speech recognition.

致謝

本篇論文的完成，首先要由衷地感謝我的指導教授張文輝老師，由於老師的細心教導與耐心指正，讓我瞭解到做研究時所應該持有的態度與精神，讓我受益良多。另外也要感謝實驗室的學長，同學以及學弟們，無論是在課業、研究或生活方面協助我解決問題，同時也要感謝陪伴我的朋友們，尤其在我準備研究所考試以及就讀研究所其間一路上給我鼓勵並提供我做研究的動力。最後，僅將此論文獻給我最親愛的父母以及家人們。



目錄

中文摘要	i
英文摘要	ii
致謝	iii
目錄	iv
圖目錄	vii
表目錄	viii
第一章 緒論	1
1.1 研究動機與方向	1
1.2 章節概要	3
第二章 訊源冗息的分析	4
2.1 殘餘冗息	5
2.2 消息理論	7
2.3 分析討論	10
第三章 訊源控制通道解碼機制	15
3.1 傳輸系統	16

3.2	迴旋編碼器的架構	18
3.3	訊源控制通道解碼	21
3.3.1	基礎概念	21
3.3.2	基於符號的 BCJR 演算法	23
3.3.3	MAP 解碼器	25
第四章	符號層級 BCJR 演算法	26
4.1	前向機率的推導	26
4.2	後向機率的推導	28
4.3	機率 γ 的推導	29
4.4	前向機率與後向機率初始值的設定	31
第五章	實驗模擬與結果分析	35
5.1	基於符號的軟性輸出解碼模擬	36
5.1.1	系統模擬之步驟說明	36
5.1.2	結果分析	39
5.2	分散式語音辨識的應用	40
5.2.1	通道錯誤緩和機制	40
5.2.2	實驗模擬	44

5.2.3 結果分析.....	46
第六章 結論與未來展望.....	47
6.1 結論.....	47
6.2 未來展望.....	48
參考文獻.....	50



圖目錄

圖 2.1	框架內關聯性與框架間關聯性.....	9
圖 3.1	基本訊號傳輸模型.....	18
圖 3.2	迴旋編碼器架構.....	21
圖 3.3	訊源控制通道解碼架構的格子圖.....	22
圖 4.1	前向機率計算的柵狀圖.....	27
圖 4.2	後向機率計算的柵狀圖.....	29
圖 5.1	實驗流程圖.....	38
圖 5.2	基於符號的軟性輸出解碼之系統模擬.....	39
圖 5.3	DSR 格子架構圖.....	43
圖 5.4	輸出軟性通道解碼應用於 DSR 的模擬.....	45

表目錄

表 2.1	框架內殘餘冗息 ($M_d = 1$)	12
表 2.2	框架內殘餘冗息 ($M_d = 2$)	12
表 2.3	框架內殘餘冗息 ($M_d = 3$)	12
表 2.4	框架間殘餘冗息 ($M_d = 1$)	13
表 2.5	框架間殘餘冗息 ($M_d = 2$)	14
表 2.6	框架間殘餘冗息 ($M_d = 3$)	14
表 2.7	框架間殘餘冗息 ($M_d = 6$)	14
表 5.1	多重框架的格式	43
表 5.2	受 CRC 保護的封包串	43



第一章 緒論

1.1 研究動機與方向

在典型的數位通訊系統中，由於通道的多變性以及環境干擾的影響，訊號在接收端往往會因為嚴重失真或是衰退現象嚴重而失去原訊號所挾帶的資訊，尤其是語音 (speech)、音訊 (audio) 或是影像 (video) 的多媒體訊號受到通道環境的影響極大。有鑑於此，多媒體傳輸一直是一項很重要的研究主題。為了提升對抗通道雜訊的強健性能，我們通常會運用一些編碼工作，不論是訊源編碼 (source coding) 抑或是通道編碼 (channel coding)，主要目標就是希望能對傳輸訊號加以保護或是提供接收端更多的訊息以增加解碼的正確性。

在傳統的通訊系統設計中，訊源編碼和通道編碼這兩個系統區塊往往是分開考量且獨立設計，此設計概念源自於沈農 (shannon) 的消息理論[1]。相關理論是在個別的編碼架構規劃時，假設另一個編碼器已完成最佳化設計，但是這並不符合真實的通訊環境。經由消息理論的計算可以發現，訊號源在經由訊源編碼器的處理之後，無法完全去除輸出訊號之間的關聯性，也就是輸出訊號之間彼此並非獨立且

相同分佈。其編碼後的位元序列之間存在著某種形式的非均勻機率分佈或隱含記憶性，而這些殘留的資訊稱為殘餘冗息 (residual redundancy)。而通道編碼的主要精神是對訊號經由訊源編碼器之後的位元串多加一些保護位元，最常見的是區段碼 (block code) 以及迴旋碼 (convolutional code)。在受到通道環境的錯誤之下，這些保護位元可以在解碼工作之中降低解碼錯誤以達到保護的效果。而殘餘冗息的最大幫助就是我們可以在不需要提升通道頻寬的情形之下，提供通道編碼一些事前資訊 (a priori information) 幫助解碼工作的正確性。因此，若我們能在接收端妥善地利用這些事前資訊，相信應能對系統的解碼效能有所提升。而這些在接收端將訊源解碼器和通道環境之效應一併納入架構設計考量的研究稱為合併訊源通道解碼 (joint source-channel decoding, JSCD) [2][3][4]，而在通道解碼部分充分地利用殘餘冗息資訊的相關研究又稱為訊號源控制通道解碼 (source-controlled channel decoding) [5]。

而我們所要提出的訊號源控制通道解碼的主軸將以 BCJR 演算法的概念做延伸，在後面的章節將會有詳細的介紹。下一節為本論文的章節介紹。

另外，在行動通訊上，因為分散式語音辨識系統 (distributed speech recognition, DSR) [6] 的使用日漸增加，所以我們在實驗模

擬部分也將我們提出的軟性輸出通道解碼機制應用於此系統的接收端，並探討與原系統中所訂定的錯誤緩和機制（error mitigation）作分析比較。

1.2 章節概要

第二章先做訊號源經由訊源編碼器之後的量化輸出符號殘餘冗息分析。第三章介紹我們所探討的訊源控制通道解碼機制以及基於符號的 BCJR 演算法。第四章詳細描述基於符號 BCJR 演算法中的遞迴機率公式推導。第五章則為實驗模擬，模擬前面兩章所介紹的軟性輸出通道解碼機制，並將此機制應用於歐洲電信標準局所制訂的分散式語音辨認系統的接收端，並為模擬此架構下之結果作效能評估與比較。第六章為結論與未來展望。

第二章 訊源冗息的分析

理想的訊號源編碼器應該會在一已定的失真情況下，壓縮一連串連續的參數，並且完全移除其冗息。果真如此，編碼後的參數應當呈現均勻的機率分佈，而且每一個參數將與其鄰近的參數在時間上沒有任何的關聯性，也就是說，沒有記憶性。然而，在大多數真實的情況下，受限於延遲以及對於訊號統計結果的資訊瞭解不夠，將導致次佳的訊號源編碼器。其結果反應於訊源編碼器無法完全去除訊號源之間的冗息，而在訊源編碼器之後存在於參數之間的冗息即稱為殘餘冗息。冗息可能存在於數值的非均勻機率分佈或是時間的記憶性之中，大部分是兩者都有，尤其是以框架為基礎的語音參數，意即訊號源和時間有直接的關係。

本論文想要提出的是一軟性輸出通道解碼機制，而解碼效能所依靠的資訊將為訊源編碼輸出位元串之間所存在的殘餘冗息。可想而知的是如果我們對於殘餘冗息有深入瞭解，將對之後要提出的解碼機制能否改善其效能必定會是很大的助益。所以接下來將要討論殘餘冗息的概念，並探討此資訊對於系統效能的幫助。

2.1 殘餘冗息

當訊號源通過編碼器（或量化器）時，對於通道錯誤的敏感度將增大，所以需要使用某些技術以更正傳輸所衍生的位元錯誤。標準的通道編碼技術藉由加入了一些保護位元於編碼位元序列中，以致當位元因通道環境的影響而產生錯誤下仍可被復原。訊源冗息的基本概念為在訊源編碼器的輸出序列中，訊號源之間存在一些冗餘訊息，我們選擇不移除這些冗餘訊息，並且在通道解碼的部分利用這些資訊來提供錯誤更正。事實上，如果某些冗息被原封不動的保留下來，那麼在索引（index）分割出來的符號（symbol）之間所存在的資訊將散佈於壓縮序列中的數個符號之間。換句話說，如果雜訊使得這些符號之間的任一個產生錯誤，那麼關於此符號的資訊可以由觀察其相鄰的符號獲得。這就是在通道編碼之中，更正錯誤的方法。接下來就是為什麼我們要討論訊餘冗息的三大原因：

[1] 殘餘冗息總是存在於編碼後的符號串之間

就像之前所討論的，殘餘冗息總是存在於編碼後的符號之間。對於一些多媒體訊號源，如語音、音訊、視訊，殘餘冗息在經由訊號源編碼之後是非常值得注意的，這是因為這些訊號源總是呈現高度地不穩定，所以很難完全地壓縮。即使採用一些高壓縮率的語音編碼器，例如混和激發線性預估（the mixed-excitation linear prediction,

MELP) 訊源編碼器，仍有大量的冗息出現在編碼後的參數之間。

[2] 不需要更大的傳輸頻寬

有別於前向錯誤更正碼(forward error correcting code, FEC)[7]，利用殘餘冗息對抗通道雜訊不需要增加任何多餘的保護位元，亦即在傳輸頻寬的要求上不需要再多做增加，這在有限頻寬的通訊系統上是一項很大的優點。在[8]，Sayood *et al.* 模擬了一個考慮預估錯誤(prediction error)之間的殘餘冗息訊源解碼機制。預估錯誤序列被模擬為一階馬可夫過程，相鄰訊號源之相關性表示為訊源轉移機率。再利用通道轉移機率將通道的狀態資訊納入考量，這兩個機率組合而為一個正比於接收序列之可能性的計量(metric)，解碼的演算法相似於Viterbi演算法。

[3] 當通道雜訊很嚴重時，殘餘冗息將提供解碼所需的主要資訊

如同在[5]，訊源控制通道解碼器(Source-controlled channel decoding, SCCD)的判斷主要是依據通道的輸出以及訊號源的事前資訊。當通道環境良好，通道解碼器大部分依賴接收的序列，當通道環境惡劣時，衰退現象非常嚴重，解碼時的參考將選擇依賴訊號源的事前消息甚過於所接收的序列。值得注意的是，在極差的通道環境將使

得通道解碼喪失其錯誤更正能力，甚至還會再引進更多錯誤出現。然而，殘餘冗息在錯誤緩和（error concealment）的應用在惡劣通道環境上扮演一個很重要的角色，這觀念在現在的行動通訊系統上也是很常使用。在[5]，將 SCCD 應用於 GSM 的語音編碼上將提升 2dB 到 3dB 的改善，顯著的改善亦可在[2]中發現。

經由以上的介紹，可以知道殘餘冗息在現今或未來的傳輸系統中，在解碼部分將是一項重要的技術。在下面的小節之中，我們將分析不同程度的殘餘冗息，並在第五章的實驗模擬中證實了我們分析所得的假設。



2.2 消息理論

在探討殘餘冗息在接收端是否確實能提昇系統的解碼效能之前，我們必須先討論訊號源在經由訊源編碼處理之後，是否真的有殘餘冗息的存在。對於語音編碼器而言，在量化後的參數與參數之間往往存在大量的殘餘冗息，這是因為語音有時變的特性，所以很難完全地做壓縮。而這又取決於系統設計者是否要使用複雜先進的編碼技術大量地壓縮資料，或僅僅使用較簡單的量化器而餘留一些冗息於編碼後的位元串之間。接下來，我們將分析訊號在編碼之後所存在的殘餘冗息。

假設在 l 時刻，一 M 個位元的量化輸出索引所屬位元集合為 $U_l^{(k)}$ ，其中 $k \in \{0, 1, \dots, M-1\}$ 。我們想要探討的是不同符號 (symbol) 層級中所存在的殘餘冗息，所以在 M 個位元中，依照 M 的最大公因數切割為不同長度的符號，再計算其相鄰框架間的殘餘冗息，為了標示的方便，接下來我們將以 U_t 代表在一連串的輸出索引中，以一個符號為 M_d 位元做切割之後所有第 t 個符號的位元集合，且 $t \in \{0, 1, \dots, M/M_d - 1\}$ 。若假設每個符號之間互相獨立，則第 t 個符號的熵值 (entropy) 定義為：

$$H_t^{uni} = H(U_t) = -\sum_{u_t} P(u_t) \log_2 P(u_t) \quad (2.1)$$

接下來，定義 \mathbf{U}_m 為一個大小與索引位元個數相同的框架，因為假設每個符號為 M_d 個位元，所以此框架包含了 M/M_d 個符號，則每個框架的熵值 H_F 為：

$$H_F = \lim_{m \rightarrow \infty} \frac{1}{m} H(\mathbf{U}_0, \mathbf{U}_1, \dots, \mathbf{U}_{m-1}) \quad (2.2)$$

其中 $\mathbf{U}_m = \{u_{m,0}, u_{m,1}, \dots, u_{m,M/M_d-1}\}$ 。 H_F 意為代表一個框架所需的最少位元數，那麼符號中每一位元的平均熵值為：

$$H_{ave} = \frac{H_F}{M_d} \quad (2.3)$$

而殘餘冗息定義為：

$$\rho_t = 1 - H_{ave} \quad (2.4)$$

ρ_t 的決定因素在於符號呈現的非均勻分佈 (non-uniform

distribution) 與框架間 (inter-frame) 和框架內 (intra-frame) 的記憶性。因為上式在真實情況中是無法實現，所以 H_F 將利用一個簡單的模型作估計，這裡我們將模擬為一階馬可夫程序。

當符號間的殘餘冗息不等於0時，表示不同符號之間有關聯性存在，進一步可以被分為框架間與框架內的關聯性。框架間關聯性定義為在連續兩個相鄰框架內，不同框架中相同位置的符號之間所存在的相關性，而框架內關聯性定義為在同一個框架內相鄰兩個符號之間的相關性，如圖 2.1 所示。

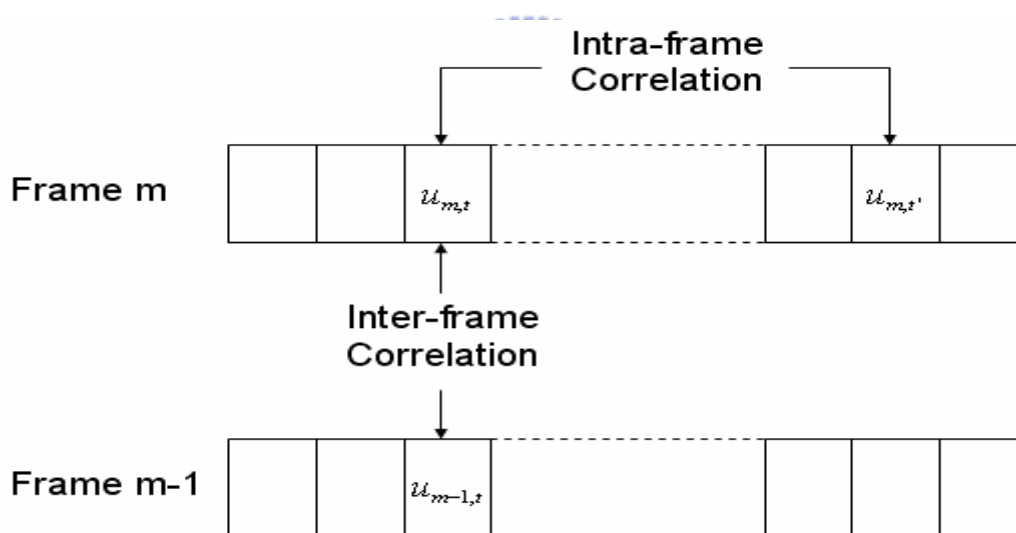


圖 2.1 框架內關聯性與框架間關聯性

如果只考慮框架間關聯性，且將符號的統計特性模擬為一階馬可夫模型，那麼第 t 個符號的熵為

$$H_t = H(U_{m,t} | U_{m-1,t}) = - \sum_{u_{m,t}} \sum_{u_{m-1,t}} P(u_{m,t}, u_{m-1,t}) \log_2 P(u_{m,t} | u_{m-1,t}) \quad (2.5)$$

相對應的殘餘冗息為

$$\rho_t = 1 - \frac{H_t}{M_d} \quad (2.6)$$

其中 $0 \leq \rho_t \leq 1$ 。此值其實包含第 t 個符號的非均勻分佈與相鄰框架內相同符號的關聯性，非均勻分佈的部分如 4.1 式為

$$\rho_t^{uni} = 1 - \frac{H_t^{uni}}{M_d} \quad (2.7)$$

而關聯性的部分為

$$\rho_t^{mem} = H_t^{uni} - H_t \quad (2.8)$$

可以輕易地導出

$$\rho_t = \rho_t^{uni} + \rho_t^{mem} \quad (2.9)$$

利用相同的推導與假設，但是考慮的是框架內相關性，可以得到第 t 個符號的熵值為

$$H_t = H(U_{m,t} | U_{m,t'}) = - \sum_{u_{m,t}} \sum_{u_{m,t'}} P(u_{m,t}, u_{m,t'}) \log_2 P(u_{m,t} | u_{m,t'}) \quad (2.10)$$

為了簡單起見，我們考慮相同框架中相鄰符號之間的關聯性，即 $t' = t - 1$ 。

2.3 分析討論

接下來，我們將用上述的討論來分析一階自迴歸程序 (first-order autoregressive process) 的訊號 v_n ，在不同符號層

級中所存在的訊源冗息，訊號源之相關因子 (correlation factor) $\rho_{vv} = 0.9$ 且變異數為 $\sigma_v^2 = 1$ 。在訊源編碼部分，我們利用 LBG 演算法 (Linde-Buze-Gray algorithm) 訓練產生的向量量化器 (vector quantizer)，並且設定索引位元數 $M = 6$ ，索引的大小即為每個框架的大小，在產生出 1000000 筆索引之後，以索引數的最大公因數分別分割不同程度的符號 ($M_d = 1, 2, 3, 6$)，再計算不同位元數的符號之間所存在的記憶性 (框架間與框架內的關聯性) 與殘餘冗息。

殘餘冗息的計算方法如 (2.9) 式所示，接下來我們先考慮框架內記憶性的分析，再考慮框架間記憶性的分析。另外，當 $M_d = 6$ 時，即框架的大小等於索引大小，我們僅討論框架間的記憶性。我們想要觀察的是當分割為不同層級的符號，殘餘冗息 ρ_t 的變化如何。

(1) 考慮框架內記憶性

表 2.1 到表 2.3 為考慮當 $M_d = 1, 2, 3$ 的框架內記憶性，當 M_d 從 1 個位元增加到 3 個位元， ρ_t 很明顯漸漸地提升。在相同框架中，相鄰符號之間的相關性有微幅的提升，但是提升的幅度不大，也瞭解符號與符號之間確實有殘餘冗息存在，當符號位元數 M_d 增加，殘餘冗息也跟著增加。

表 2.1 框架內殘餘冗息 ($M_d = 1$)

t- symbol	ρ_t^{uni}	ρ_t^{mem}	ρ_t
1	0.003933	0	0.003933
2	0.005647	0.008822	0.014469
3	0.002140	0.002559	0.004699
4	0.005073	0.004995	0.010068
5	0.002781	0.002824	0.005605
6	0.000369	0.000616	0.000985

表 2.2 框架內殘餘冗息 ($M_d = 2$)

t- symbol	ρ_t^{uni}	ρ_t^{mem}	ρ_t
1	0.006444	0	0.006444
2	0.003630	0.007080	0.010710
3	0.001725	0.018090	0.019815

表 2.3 框架內殘餘冗息 ($M_d = 3$)

t- symbol	ρ_t^{uni}	ρ_t^{mem}	ρ_t
1	0.005744	0	0.005744
2	0.003813	0.044341	0.048154

(2) 考慮框架間記憶性

表 2.4 到表 2.7 為 $M_d = 1, 2, 3, 6$ 的框架內記憶性，從表 2.4 到 2.6，我們可以看出當 M_d 從 1 個位元到 3 個位元， ρ_t 很明顯漸漸地提升，也就是說在不同框架中但相同位置的符號之間所存在的關聯性在符號

位元數越來越大時，也相對地提升，殘餘冗息不但存在且隨著位元數 M_d 的增加而越來越多。然而，值得注意的是當符號位元數與索引位元數相等時， ρ_t 突然大幅地提升，也就是說當 $M_d = M = 6$ 時，索引與索引之間可以利用的殘餘冗息大量地增加。

在分析完以上的結果之後，我們發現系統的解碼效能在符號的位元數增大的同時，相對應的殘餘冗息也跟著增加，尤其當符號位元數與索引位元數相等時，殘餘冗息的增加最為顯著。我們在後面章節所要提出的軟性輸出通道解碼機制，其效能評斷所依靠的資訊為訊源之間的殘餘冗息，所以由本章的分析結果，我們將預見的是殘餘冗息的增加將對於系統的解碼效能賦予相對應的提升，尤其是當索引大小與符號大小相等的情況，由於殘餘冗息大量地增加，所以對於解碼的能力幫助應該會是最大。

表 2.4 框架間殘餘冗息 ($M_d = 1$)

t- symbol	ρ_t^{uni}	ρ_t^{mem}	ρ_t
1	0.003933	0.004067	0.008
2	0.005647	0.005910	0.011557
3	0.002140	0.002760	0.0049
4	0.005073	0.005119	0.010192
5	0.002781	0.002903	0.005684
6	0.000369	0.000401	0.00077

表 2.5 框架間殘餘冗息 ($M_d = 2$)

t- symbol	ρ_t^{umi}	ρ_t^{mem}	ρ_t
1	0.006444	0.002442	0.008886
2	0.003630	0.000787	0.004417
3	0.001725	0.004377	0.006103

表 2.6 框架間殘餘冗息 ($M_d = 3$)

t- symbol	ρ_t^{umi}	ρ_t^{mem}	ρ_t
1	0.005744	0.007322	0.013066
2	0.003813	0.012160	0.015973

表 2.7 框架間殘餘冗息 ($M_d = 6$)

t-symbol	ρ_t^{umi}	ρ_t^{mem}	ρ_t
1	0.025945	0.188117	0.214062

第三章 訊源控制通道解碼機制

在傳統的數位通訊系統，我們通常會加上通道編碼以期能夠提昇抵抗通道錯誤的保護能力，其中最為廣泛使用的通道編碼為迴旋碼。在解碼端，通常相對應於迴旋碼的解碼工作所使用的是 Viterbi 演算法，其缺點為僅僅只能執行硬性判定而提供最大可能性傳送序列的訊息(the most probable transmitted sequence)。有鑑於此，我們將介紹一個以軟性輸出通道解碼的概念為主軸的錯誤隱匿機制，在接收端先計算以訊源符號(symbol)為單位的後驗機率 (a posteriori probability)。雖然後來也有軟性輸出 Viterbi 演算法(SOVA)[9]被提出，但其仍屬於次佳(sub-optimal)。就我們所知，BCJR 演算法[10][11]利用前向-後向最大後驗機率將位元錯誤率降至最低值。不同於傳統基於位元的 BCJR 演算法，我們將推導符號層級的 BCJR 演算法(symbol-based BCJR algorithm)。

近來許多研究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，稱為合併訊源通道解碼。主要是考慮到訊號在通過量化處理之後，因為訊源編碼的非完美性而存在一些殘餘冗息。有鑑於此，我們可以在解碼端充分地引用這些資訊及其間的相關性，

再結合通道解碼機制以對抗通道雜訊之干擾，此即為訊源控制通道解碼。

而在本章的內容裡，我們將先介紹整個系統的傳輸模型，再依序介紹迴旋碼的編碼結構，BCJR 演算法以及最大後驗機率(Maximum a posteriori, MAP)解碼器[8][12]。

3.1 傳輸系統

圖 3.1 為數位通訊上被廣泛地使用的格子流程圖，將作為本論文的主要框架。接下來將分別簡介每個工作單元的功能。



訊源編碼/解碼器(Source Encoder/Decoder)

訊源編碼器的主要目的是壓縮訊號源以降低傳輸速率。最普遍使用的編碼模式是純量(scalar)或向量量化器(vector quantizer)，將實數表示的訊號映射到一個以 M 個位元表示的索引集合 J ， $J \in \{0, 1, \dots, 2^M - 1\}$ 。而在取樣時刻(sample time) n ，被量化的訊號源 v_n 將以一索引值 x_n 代表。在接收端，訊源解碼器將依據解碼而得的索引值 \hat{x}_n 經由量化碼書查表(codebook lookup)比對重建其訊號為 \hat{v}_t 。

位元映射/反映射(BM/BM⁻¹)

經由位元映射(bit mapping)，將每個索引 x_n 分別配置一含有 M 個位元的位元組合 $\mathbf{U}_n @ \{u_{n,0}, u_{n,1}, \dots, u_{n,M-1}\}$ 。而在接收端，解碼而得的位元組合 $\hat{\mathbf{U}}_n @ \{\hat{u}_{n,0}, \hat{u}_{n,1}, \dots, \hat{u}_{n,M-1}\}$ 將經由反位元映射得到相對應的索引 \hat{x}_i 。

通道編碼/解碼器(Channel Encoder/Decoder)

通道編碼/解碼器是通訊系統中極為必要的一個部分，因為它對於訊號源提供了對抗錯誤的保護功能。藉由增加一些多餘的保護位元到資訊位元上，不僅可以提供錯誤偵測還有對於錯誤的更正，使得傳送的資訊受到通道錯誤所產生的影響減小。假設編碼器的編碼率為 $1/N_c$ ，若今傳送 \mathbf{U}_n ，通道編碼器輸出為 $\mathbf{Y}_n @ \{y_{n,0}, y_{n,1}, \dots, y_{n,MN_c-1}\}$ ，而被通道錯誤影響後所接收到的序列 $\mathbf{Y}_n^o @ \{y_{n,0}^o, y_{n,1}^o, \dots, y_{n,MN_c-1}^o\}$ ，則通道解碼器的輸出為一近似 \mathbf{U}_n 的位元組合 $\hat{\mathbf{U}}_n$ 。

調變/解調變器(Modulator/Demodulator)

調變器主要是在用來將一離散的有限符號序列轉換為能夠適當地在實體通道上傳送的連續時間類比訊號。解調變器則是處理經由通道所接收到的訊號，在復原了從類比波形所得的數位訊號後產生數值輸出。

通道(Channel)

通道是使傳輸訊號受到干擾與雜訊的傳輸媒介，因此接收到的訊號會與傳送的訊號不一致。傳統的通道錯誤包括高斯雜訊(Gaussian noise)，多路徑衰退(multipath fading)，相對路徑延遲(relative path delay)，杜卜勒效應(Doppler shift)等。而本文則僅考慮可加性高斯白雜訊通道(AWGN channel)。

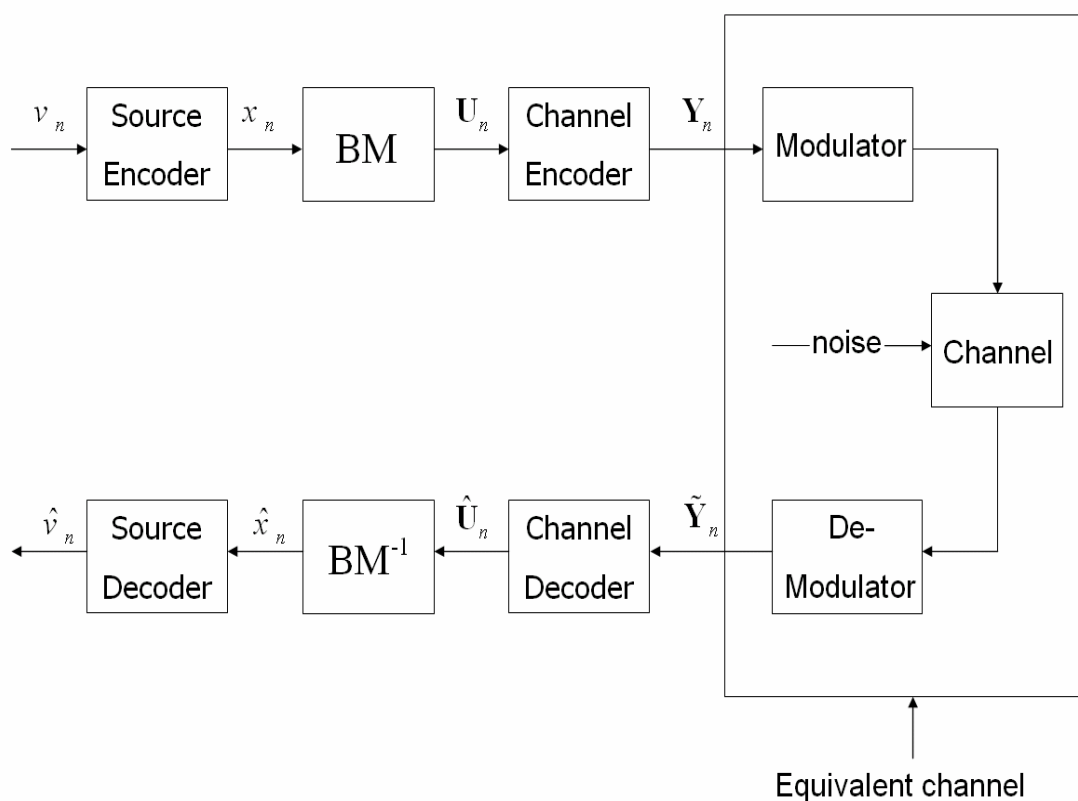


圖 3.1 基本訊號傳輸模型

3.2 迴旋編碼器的架構

考慮一個暫存記憶體為 M_c ，限制長度(constraint length)為

$M_c + 1$ ，而編碼率為 $1/N_l$ 的通道編碼器。一般而言，輸入一串二位元序列 $\{u_0, u_1, \dots, u_{N-1}\}$ 至編碼器，其輸出端會產生另一串二位元的序列 $\{y_0, y_1, \dots, y_{N-1}\}$ ，其中 N 為位元總數而 $\mathbf{y}_t = \{y_{t,0}, y_{t,1}, \dots, y_{t,N_l-1}\}$ 。有別於傳統依單一位元處理模式，我們將要探討每次一個單位時間輸入一個 M_d 位元的符號(symbol)，致使編碼器的移位暫存器在同一單位時間內也移動了 M_d 個位元。在下一段，我們將詳盡的探討此基於符號之迴旋編碼器。

在單位取樣時間 (sample time) 內，訊號源 v_n 通過量化器之後可得到一個以 M 個位元表示的索引 x_t ，在這裡，我們將此索引每 M_d 個位元做分割，即可得到 M/M_d 個符號 u_t ，其中 $t = n \cdot M_d + m$ 且 $m = 0, 1, \dots, M/M_d - 1$ ， t 為符號時間 (symbol time)。

現在設定每個符號為 M_d 個位元，並假設在量化器的輸出總共會產生 N 個位元，即編碼器的輸入為 $\{u_0, u_1, \dots, u_{N_s-1}\} = \{b_0, b_1, \dots, b_{N-1}\}$ ，其中 $u_t = \{b_{t \times M_d}, b_{t \times M_d + 1}, \dots, b_{t \times M_d + M_d - 1}\}$ 且符號總數為 $N_s = N/M_d$ 。輸出序列為 $\{\mathbf{Y}_0, \mathbf{Y}_1, \dots, \mathbf{Y}_{N_s-1}\}$ ，其中 $\mathbf{Y}_t = \{y_0, y_1, \dots, y_{M_d-1}\} = \{y_0, y_1, \dots, y_{M_d N_l - 1}\}$ 且 $\mathbf{y}_l = \{y_{l,0}, y_{l,1}, \dots, y_{l,N_l-1}\}$ 。

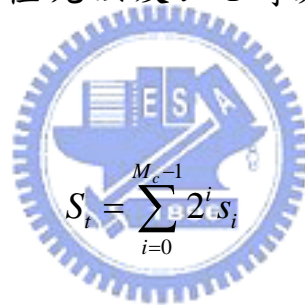
編碼器的產生多項式(encoder generator polynomials)定義為：

$$G_j(D) = g_0^j + Dg_1^j + D^2g_2^j + \dots + D^{M_c}g_{M_c}^j, \quad j = 0, 1, \dots, N_l - 1 \quad (3.1)$$

而對個別位元 $y_{l,j}, l = 0, 1, \dots, M_d - 1, j = 0, 1, \dots, N_l - 1$ 而言，

$$y_{l,j} = b_l g_0^j + \sum_{k=1}^{M_c} g_k^j s_k \pmod{2}, g_k^j \in \{0,1\} \quad (3.2)$$

s_k 為移位暫存器 (shift register) 中所儲存的位元值，其中 $k=0,1,\dots,M_c$ ，而編碼器的狀態值 S_t 由 s_k 所決定，如(3.3)式。值得注意的是，暫存器的狀態值是由每個單位時間內輸入符號的位元組合完全進入暫存器之後計算得到。 S_t 將由最近 M_c 個輸入位元決定，其中若 $M_d > M_c$ ，則由單位時間內輸入至編碼器的最後 M_c 個位元決定，前 $M_d - M_c$ 個位元將不影響狀態值 S_t 。反之，若 $M_d < M_c$ ，則 S_t 將由 t 時刻輸入的 M_c 位元以及 $t-1$ 時刻所輸入至編碼器的後 $M_c - M_d$ 位元決定。



$$S_t = \sum_{i=0}^{M_c-1} 2^i s_i \quad (3.3)$$

由(3.3)式可得，此暫存器共可產生 2^{M_c} 種可能出現的狀態。另外，迴旋編碼器的起始狀態 $S_0 = 0$ 需設定為0。全部的輸入序列傳送完畢之後，必須額外傳送 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。

一個典型迴旋編碼器架構圖如圖 3.2 所示，此編碼器的限制長度為5，編碼率為1/2，產生多項式為

$$G_1(D) = 1 + D^3 + D^4 \quad (3.4)$$

$$G_2(D) = 1 + D^1 + D^3 + D^4 \quad (3.5)$$

此編碼架構將為第五章實驗模擬系統的編碼器。

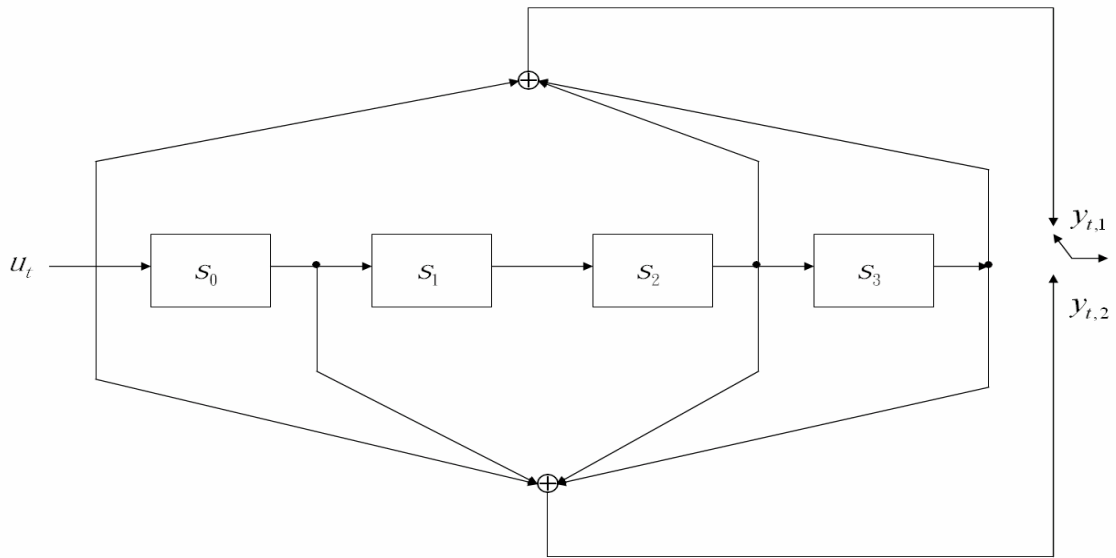


圖 3.2 迴旋編碼器架構

3.3 訊源控制通道解碼

3.3.1 基礎概念

標準的 SCCD 架構如圖 3.3 所示。SCCD 和一般傳統的通道解碼架構的不同之處，在於除了接收訊號 \mathcal{Y}_T 之外，通道解碼器的輸入端還需要另一項資訊-訊源的事前資訊(a priori information)。事前資訊的最大好處來自通道環境惡劣時，通道解碼端將依賴符號值的事前資訊甚過於收到的符號值。反之，當訊號通過一良好的通道環境時，通道解碼器將大部分依賴接收到的符號值。這是因為在非常惡劣的通道下，接收到的符號將由於太多的通道錯誤使得錯誤更正機制很難有效地正確解碼。而最常被使用的事前資訊，是根據訊號源量化符號的機率分佈作分析，可以由大量的訓練序列而得。

訊號在經過訊源編碼之後，被壓縮處理的語音參數對於錯誤的抵抗能力原本就很薄弱，尤其當通道環境極差的時候，在遭受無可更正的錯誤之後，系統效能將會嚴重地下降。在這種情況之下，SCCD 將會是一個針對接收序列，在不增加傳輸頻寬的情況下而能夠大大地改善解碼品質的有效方法。在經由編碼之後，利用相鄰音框之間的相關性，通道解碼器可以計算出所有可能符號值的後驗機率，換句話說，就是軟性輸出解碼器(soft-output decoder)。接下來，我們將要提出一個 SCCD，其分為兩個子系統，第一個子系統介紹了以符號為基礎的 BCJR 演算法來當作此迴旋碼的解碼器計算得到一軟性輸出；第二個子系統跟隨著一個最大後驗機率解碼器(maximum a posteriori probability MAP decoder)，利用所接收到的軟性輸出作估測。

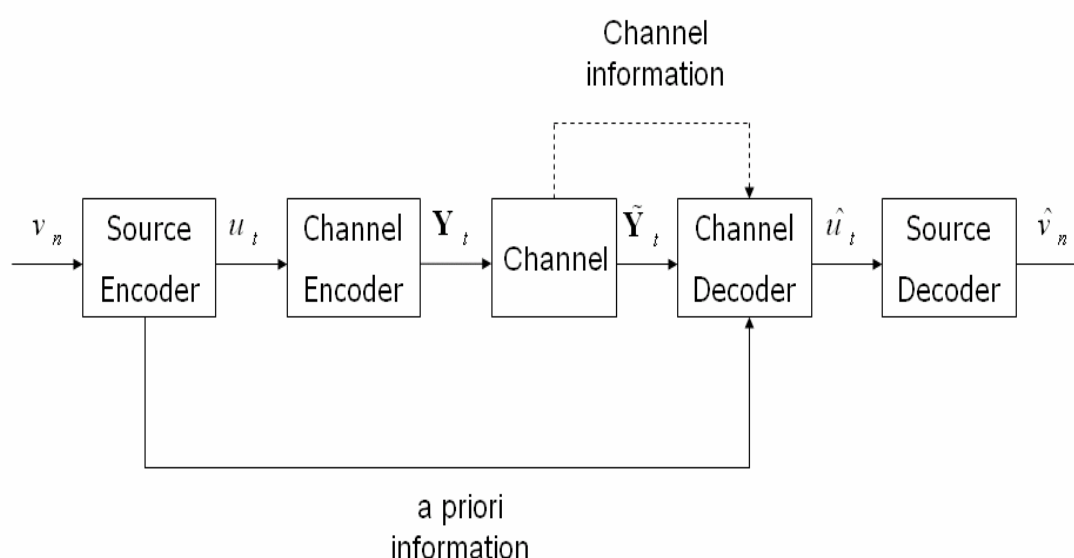


圖 3.3 訊源控制通道解碼架構的格子圖

3.3.2 基於符號的 BCJR 演算法

依照通訊系統裡的通道編碼器架構如 3.2 節所提及，假設以 N_s 個符號為格子架構裡一個主要的消息區塊且每一個符號為 M_d bit 的位元組合，經由通道編碼器傳送至可加性白色高斯雜訊通道中，由於通道的輸入 $y_{t,l}$ 與通道的輸出 $\mathcal{Y}_{\rho,l}$ 受到雜訊 $n_{t,l}$ 的影響，使得在接收端，通道解碼器將收到實數序列 $\mathbf{Y}_0^{\mathcal{Y}_{N_s-1}}$ $@(\mathbf{Y}_0^{\mathcal{Y}_0}, \mathbf{Y}_1^{\mathcal{Y}_K}, \mathbf{Y}_{N_s-1}^{\mathcal{Y}_{N_s-1}})$ ，其中

$\mathbf{Y}_t^{\mathcal{Y}_t} @\{\mathcal{Y}_{\rho,0}, \mathcal{Y}_{\rho,1}, K, \mathcal{Y}_{\rho, M_d N_t - 1}\}$ ，且

$$\mathcal{Y}_{\rho,l} = y_{t,l} + n_{t,l}, l = 0, 1, K, M_d N_t - 1 \quad (3.6)$$

其中 $n_{t,l}$ 為零平均值，且變異數為 σ^2 的高斯雜訊取樣。

在接收端，通道解碼器將根據所收到的序列 $\mathbf{Y}_0^{\mathcal{Y}_{N_s-1}}$ 找出最可能傳送的符號 u_t 。計算每一個解碼後符號 u_t 的後驗機率為

$$P[u_t | \mathbf{Y}_0^{\mathcal{Y}_{N_s-1}}] = \sum_{s_t} \lambda_t^i(s_t) \quad (3.7)$$

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

$$\lambda_t^i(s_t) = P(u_t = i, S_t = s_t | \mathbf{Y}_0^{\mathcal{Y}_{N_s-1}}), i \in \{0, 1, K, 2^{M_d-1}\} \quad (3.8)$$

藉由貝式定理，可得：

$$\begin{aligned}
\lambda_t^i(s_t) &= P(u_t = i, S_t = s_t | \mathbf{Y}_0^{\mathcal{G}^{N_s-1}}) \\
&= \frac{P(u_t = i, S_t = s_t, \mathbf{Y}_0^{\mathcal{G}^{N_s-1}})}{P(\mathbf{Y}_0^{\mathcal{G}^{N_s-1}})} \\
&= \frac{P(u_t = i, S_t = s_t, \mathbf{Y}_0^{\mathcal{G}^{N_s-1}})}{\sum_i \sum_{s_t} P(u_t = i, S_t = s_t, \mathbf{Y}_0^{\mathcal{G}^{N_s-1}})} \\
&= \frac{P(u_t = i, S_t = s_t, \mathbf{Y}_0^{\mathcal{G}}, \mathbf{Y}_{t+1}^{\mathcal{G}^{N_s-1}})}{\sum_i \sum_{s_t} P(u_t = i, S_t = s_t, \mathbf{Y}_0^{\mathcal{G}^{N_s-1}})} \\
&= \frac{P(u_t = i, S_t = s_t, \mathbf{Y}_0^{\mathcal{G}}) P(\mathbf{Y}_{t+1}^{\mathcal{G}^{N_s-1}} | u_t = i, S_t = s_t, \mathbf{Y}_0^{\mathcal{G}})}{\sum_i \sum_{s_t} P(u_t = i, S_t = s_t, \mathbf{Y}_0^{\mathcal{G}^{N_s-1}})} \\
&= \frac{\alpha_t^i(s_t) \beta_t^i(s_t)}{\sum_i \sum_{s_t} \alpha_t^i(s_t) \beta_t^i(s_t)} \tag{3.9}
\end{aligned}$$

其中 α 和 β 的定義為

$$\alpha_t^i(s_t) = P(u_t = i, S_t = s_t, \mathbf{Y}_0^{\mathcal{G}}) \tag{3.10}$$

$$\beta_t^i(s_t) = P(\mathbf{Y}_{t+1}^{\mathcal{G}^{N_s-1}} | u_t = i, S_t = s_t) \tag{3.11}$$

分別被稱為前向(forward)機率和後向(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，降低運算量以便於計算，其型式將如下所示：

$$\alpha_t^i(s_t) = \sum_{s_{t-1}} \sum_j \alpha_{t-1}^j(s_{t-1}) \gamma_{i,j}(\mathbf{Y}_0^{\mathcal{G}}, s_{t-1}, s_t) \tag{3.12}$$

和

$$\beta_t^i(s_t) = \sum_{s_{t+1}} \sum_j \beta_{t+1}^j(s_{t+1}) \gamma_{j,i}(\mathbf{Y}_0^{\mathcal{G}}, s_{t+1}, s_t) \tag{3.13}$$

其中上式的 α, β, γ 將在下一章做詳細的推導與分析。

3.3.3 MAP 解碼器

一般最常見的兩種估測方法為最小均方錯誤解碼器 (MMSE decoder) [13][14]與最大後驗機率解碼器。對於一個離散系統，MMSE 解碼器通常被認為是減小原始的與估計的符號值之間的均方誤差的最佳方法，而 MAP 解碼器則是以解碼的位元錯誤率 (bit error rate, BER) 降至最低為最佳。本論文中我們將使用的是 MAP 解碼器。

在接收端收到序列 $\mathbf{Y}_0^{N_s-1}$ 後，經由上一節的後驗機率計算後，我們將會得到 N_s 個軟性輸出 $P(u_t | \mathbf{Y}_0^{N_s-1})$ ，再利用最大後驗機率的估計準則 (3.15) 式，可估測得到符號值 \hat{u}_t ，並再依照在傳送端分解索引 x_n 的步驟回復，在每個取樣時間下，將每 M/M_d 個符號 \hat{u}_t 組合回來，即可得到估計的索引值 \hat{x}_n 。在得到 \hat{x}_n 之後，再利用量化碼書查表比對查詢可得 \hat{v}_n 。

MAP decoder:

$$\hat{u}_t = \arg \max_{u_t} P(u_t | \mathbf{Y}_0^{N_s-1}) \quad (3.15)$$

第四章 符號層級 BCJR 演算法

目前 SCCD 的相關研究中，大都是整合位元層級的事前資訊於 BCJR 演算法(bit-based BCJR algorithm)架構。而在 3.3.2 節中，我們推導一基於符號的 BCJR 演算法，雖然複雜度增加，但在第二章已證明相鄰符號間存在更多的殘餘冗息資訊可以在解碼端被充分地利用。而在接下來的四個小節中，我們將詳盡地推導(3.9)，(3.12)，(3.13) 中的遞迴機率 α, β, γ ，並瞭解其初始值設定的重要性對系統效能有極大的影響。



4.1 前向機率的推導

前向機率的定義為

$$\alpha_t^i(s_t) = P(u_t = i, S_t = s_t, \mathbf{Y}_0^t), i \in \{0, 1, K, 2^{M_d} - 1\} \quad (4.1)$$

依貝氏定理展開為

$$\begin{aligned} \alpha_t^i(s_t) &= P(u_t = i, S_t = s_t, \mathbf{Y}_0^{t-1}, \mathbf{Y}_t^t) \\ &= \sum_{s_{t-1}} \sum_j P(u_t = i, u_{t-1} = j, S_t = s_t, S_{t-1} = s_{t-1}, \mathbf{Y}_0^{t-1}, \mathbf{Y}_t^t) \\ &= \sum_{s_{t-1}} \sum_j P(u_t = i, S_t = s_t, \mathbf{Y}_t^t | u_{t-1} = j, S_{t-1} = s_{t-1}, \mathbf{Y}_0^{t-1}) \\ &\quad \times P(u_{t-1} = j, S_{t-1} = s_{t-1}, \mathbf{Y}_0^{t-1}) \end{aligned} \quad (4.2)$$

根據前向機率的定義，在上式，首先我們可以得到

$$P(u_{t-1} = j, S_{t-1} = s_{t-1}, \mathbf{Y}_0^{t-1}) = \alpha_{t-1}^j(s_{t-1}) \quad (4.3)$$

另外，因為從 s_{t-1} 到 s_t 的轉移路徑是由 u_{t-1} 和 s_{t-1} 決定，和接收到的序列 \mathbf{Y}_0^{t-1} 沒有任何的關係，所以

$$\begin{aligned} & P(u_t = i, S_t = s_t, \mathbf{Y}_t^0 | u_{t-1} = j, S_{t-1} = s_{t-1}, \mathbf{Y}_0^{t-1}) \\ &= P(u_t = i, S_t = s_t, \mathbf{Y}_t^0 | u_{t-1} = j, S_{t-1} = s_{t-1}) \\ & @ \gamma_{i,j}(\mathbf{Y}_t^0, s_{t-1}, s_t) \end{aligned} \quad (4.4)$$

所以 (4.1) 式可簡化成以下的前向遞迴計算型式：

$$\alpha_t^i(s_t) = \sum_{s_{t-1}} \sum_j \alpha_{t-1}^j(s_{t-1}) \gamma_{i,j}(\mathbf{Y}_t^0, s_{t-1}, s_t) \quad (4.5)$$

前向機率遞迴計算所對應的柵狀圖(trellis diagram)如圖 4.1 所

示，假設 $M_c = 4$ 且 $M_d = 2$ 。

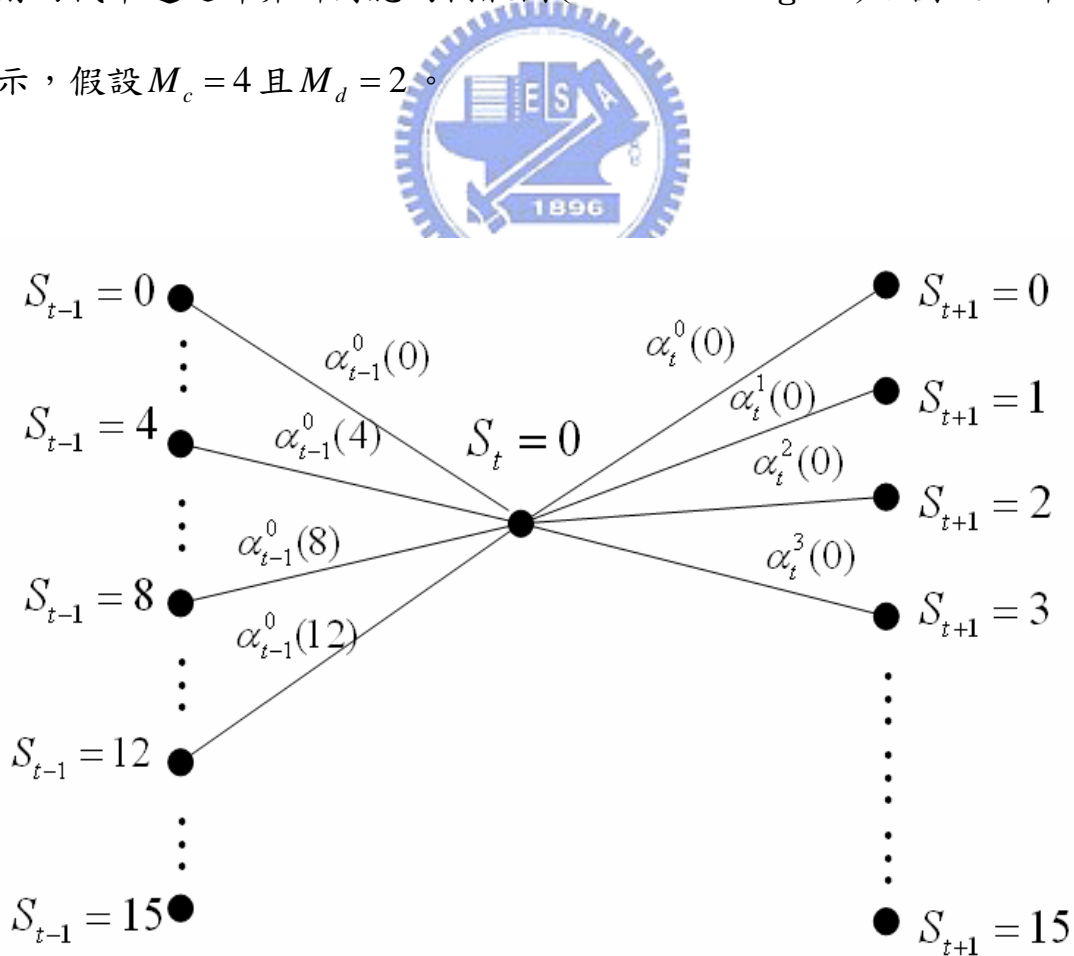


圖 4.1 前向機率計算的柵狀圖

4.2 後向機率的推導

後向機率的定義為

$$\beta_t^i(s_t) = P(\mathbf{Y}_{t+1}^{\mathcal{O}^{N_s-1}} | u_t = i, S_t = s_t), i \in \{0, 1, \dots, K, 2^{M_d} - 1\} \quad (4.6)$$

其遞迴計算的推導和 $\alpha_t^i(s_t)$ 相似，但必須在整個序列接收完全後，才可以開始計算，(4.6) 式可寫成

$$\begin{aligned} \beta_t^i(s_t) &= P(\mathbf{Y}_{t+2}^{\mathcal{O}^{N_s-1}}, \mathbf{Y}_{t+1}^{\mathcal{O}} | u_t = i, S_t = s_t) \\ &= \sum_{s_{t+1}} \sum_j P(u_{t+1} = j, S_{t+1} = s_{t+1}, \mathbf{Y}_{t+2}^{\mathcal{O}^{N_s-1}}, \mathbf{Y}_{t+1}^{\mathcal{O}} | u_t = i, S_t = s_t) \\ &= \sum_{s_{t+1}} \sum_j P(\mathbf{Y}_{t+2}^{\mathcal{O}^{N_s-1}} | u_{t+1} = j, S_{t+1} = s_{t+1}, \mathbf{Y}_{t+1}^{\mathcal{O}}, u_t = i, S_t = s_t) \\ &\quad \times P(u_{t+1} = j, S_{t+1} = s_{t+1}, \mathbf{Y}_{t+1}^{\mathcal{O}} | u_t = i, S_t = s_t) \end{aligned} \quad (4.7)$$

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

$$\begin{aligned} &P(\mathbf{Y}_{t+2}^{\mathcal{O}^{N_s-1}} | u_{t+1} = j, S_{t+1} = s_{t+1}, \mathbf{Y}_{t+1}^{\mathcal{O}}, u_t = i, S_t = s_t) \\ &= P(\mathbf{Y}_{t+2}^{\mathcal{O}^{N_s-1}} | u_{t+1} = j, S_{t+1} = s_{t+1}) \\ &= \beta_{t+1}^j(s_{t+1}) \end{aligned} \quad (4.8)$$

若再定義

$$\gamma_{j,i}(\mathbf{Y}_{t+1}^{\mathcal{O}}, s_{t+1}, s_t) = P(u_{t+1} = j, S_{t+1} = s_{t+1}, \mathbf{Y}_{t+1}^{\mathcal{O}} | u_t = i, S_t = s_t) \quad (4.9)$$

(4.7) 式可以用後向遞迴的計算方法加以實現：

$$\beta_t^i(s_t) = \sum_{s_{t+1}} \sum_j \beta_{t+1}^j(s_{t+1}) \gamma_{j,i}(\mathbf{Y}_{t+1}^{\mathcal{O}}, s_{t+1}, s_t) \quad (4.10)$$

後向機率遞迴計算所對應的柵狀圖如圖 4.2 所示，假設 $M_c = 4$ 且

$M_d = 2$ 。

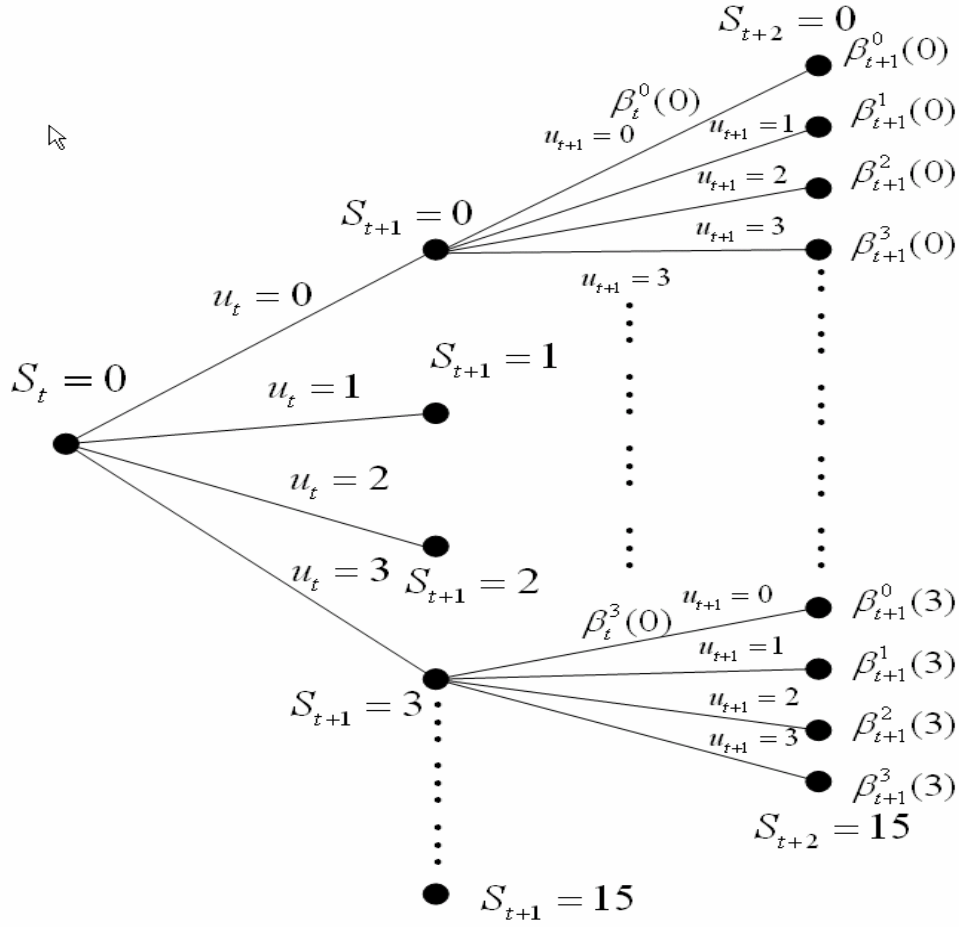


圖 4.2 後向機率計算的柵狀圖

4.3 機率 γ 的推導

機率 $\gamma_{i,j}(\mathbf{Y}_t^0, s_{t-1}, s_t)$ 可以由無記憶性高斯通道的轉移機率

(channel transition probability) 和編碼器的柵狀架構所決定。

藉著使用貝氏定理，

$$\begin{aligned}
 \gamma_{i,j}(\mathbf{Y}_t^0, s_{t-1}, s_t) &= P(u_t = i, S_t = s_t, \mathbf{Y}_t^0 | u_{t-1} = j, S_{t-1} = s_{t-1}) \\
 &= P(\mathbf{Y}_t^0 | u_t = i, S_t = s_t, u_{t-1} = j, S_{t-1} = s_{t-1}) \\
 &\quad \times P(u_t = i | S_t = s_t, u_{t-1} = j, S_{t-1} = s_{t-1}) \\
 &\quad \times P(S_t = s_t | u_{t-1} = j, S_{t-1} = s_{t-1})
 \end{aligned} \tag{4.11}$$

對一個無記憶性的高斯通道(memoryless Gaussian channel)而言，

(4.11) 式的第一項可以被寫成：

$$\begin{aligned}
 & P(\mathbf{Y}_t^0 | u_t = i, S_t = s_t, u_{t-1} = j, S_{t-1} = s_{t-1}) \\
 &= P(\mathbf{Y}_t^0 | u_t = i, S_t = s_t) \\
 &= P(\mathbf{Y}_t^0 | x(s_t, i)) \\
 &= K_t \exp\left(\frac{2}{\sigma^2} \sum_{l=0}^{M_d N_t - 1} y_{t,l} \cdot \mathcal{Y}_{t,l}^0\right) \tag{4.12}
 \end{aligned}$$

其中 K_t 為一常數，而 $x(s_t, i)$ 為代表了在給定狀態 s_t 和輸入符號值 i 之後的輸出符號。接著，考慮(4.11)式中的第二項機率，其意謂著訊號源量化後之前後相鄰兩符號間的關聯性，大致上可分為如下列的情況討論之：

(1) 若假設量化後之相鄰符號間不存在殘餘殘餘冗息，亦即彼此之間為獨立相同分佈(i. i. d.)，則(4.11)式中的第二項機率可寫為：

$$\begin{aligned}
 & P(u_t = i | S_t = s_t, u_{t-1} = j, S_{t-1} = s_{t-1}) \\
 &= P(u_t = i) = \frac{1}{2^{M_d}} \tag{4.13}
 \end{aligned}$$

(2) 倘若將符號間的殘餘冗息納入考量，也就是利用訊號經過量化器之後符號間所存在的相關性，因此可得：

$$\begin{aligned}
 & P(u_t = i | S_t = s_t, u_{t-1} = j, S_{t-1} = s_{t-1}) \\
 &= P(u_t = i | u_{t-1} = j) \tag{4.14}
 \end{aligned}$$

而此機率值意謂著序列中的相鄰符號之間的關聯性可模擬為一階馬可夫程序 (first-order Markov process, AK1)，可經由事前經大量

語料訓練而得。

在 (4.11) 式中的第三項機率為：

$$P(S_t = s_t | u_{t-1} = j, S_{t-1} = s_{t-1}) = \begin{cases} 1, & \text{if } s_{t-1} = s_b^j(s_t) \\ 0, & \text{otherwise} \end{cases} \quad (4.15)$$

其中 $s_b^j(s_t)$ 為給定了由 $S_t = s_t$ 和 $u_{t-1} = j$ 所定義的路徑之後，所得出 s_t 的前一個狀態。此項機率決定了柵狀架構中，相鄰兩時間點之間狀態的轉移和路徑的行進方向。在 (4.9) 式中的 $\gamma_{j,i}(\mathbf{Y}_{t+1}^o, s_{t+1}, s_t)$ 如同 (4.4) 中的 $\gamma_{i,j}(\mathbf{Y}_t^o, s_t, s_{t-1})$ 可依照類似於如上所描述的方法而求得。

4.4 前向機率和後向機率初始值的設定

由於我們的系統是架構於迴旋編碼的基礎上，而迴旋編碼的主要精髓又在於輸入 u_t 與狀態 s_t 之間的相互關係所產生的柵狀架構圖。因此對於柵狀架構圖的路徑的瞭解，將有助於我們對前向機率與後向機率初始值的設定。然而，此設定對解碼程序的影響極大，因為初始值的設定主要是源自於迴旋編碼過程中柵狀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尾巴位元迫使結束狀態回歸到零的相關條件。把這些限制合理地融入演算法內，拒絕了不可能的發生路徑與狀態，將會是維持解碼程序之可靠性上很重要的一環。

前向機率之初始定義為 $\alpha_0^i = P(u_0 = i, S_0 = s_0, \mathbf{Y}_0^o)$ ，透過條件機率的

展開以及初始狀態必為零的限制，可得其設定為

$$\begin{aligned}\alpha_0^i &= P(\mathcal{Y}_0^i | u_0 = i, S_0 = s_0)P(u_0 = i | S_0 = s_0)P(S_0 = s_0) \\ &= \begin{cases} P(\mathcal{Y}_0^i | u_0 = i, S_0 = 0)(u_0 = i) & \text{if } s_0 = 0 \\ 0 & \text{if } s_0 \neq 0 \end{cases} \end{aligned} \quad (4.16)$$

上述亦即排除了初始狀態不為零的事件。

接下來考慮後向機率的初始狀態。首先，在時刻 $t = N_s - 1$ 時先計算後驗機率 $\lambda_{N_s-1}^i(s_{N_s-1}) = P(u_{N_s-1}, S_{N_s-1} | \mathcal{Y}_0^{N_s-1})$ ，我們將此後驗機率利用貝氏定理展再加上 $\alpha_{N_s-1}^i(s_{N_s-1})$ 的定義可得

$$\begin{aligned}\lambda_{N_s-1}^i(s_{N_s-1}) &= P(u_{N_s-1} = i, S_{N_s-1} = s_{N_s-1} | \mathcal{Y}_0^{N_s-1}) \\ &= \frac{P(u_{N_s-1} = i, S_{N_s-1} = s_{N_s-1}, \mathcal{Y}_0^{N_s-1})}{\sum_i \sum_{s_{N_s-1}} P(u_{N_s-1} = i, S_{N_s-1} = s_{N_s-1}, \mathcal{Y}_0^{N_s-1})} \\ &= \frac{P(u_{N_s-1} = i, S_{N_s-1} = s_{N_s-1}, \mathcal{Y}_0^{N_s-1})}{\sum_i \sum_{s_{N_s-1}} P(u_{N_s-1} = i, S_{N_s-1} = s_{N_s-1}, \mathcal{Y}_0^{N_s-1})} \\ &= \frac{\alpha_{N_s-1}^i(s_{N_s-1})}{\sum_i \sum_{s_{N_s-1}} \alpha_{N_s-1}^i(s_{N_s-1})} \end{aligned} \quad (4.17)$$

再與(3.9)式做比較，可得在 $t = N_s - 1$ 時刻，

$$\frac{\alpha_{N_s-1}^i(s_{N_s-1})}{\sum_i \sum_{s_{N_s-1}} \alpha_{N_s-1}^i(s_{N_s-1})} = \frac{\alpha_{N_s-1}^i(s_{N_s-1})\beta_{N_s-1}^i(s_{N_s-1})}{\sum_i \sum_{s_{N_s-1}} \alpha_{N_s-1}^i(s_{N_s-1})\beta_{N_s-1}^i(s_{N_s-1})}$$

(4.18)

故欲滿足上式等號成立，合理的設定為 $\beta_{N_s-1}^i(s_{N_s-1}) = 1$ 。同時再探討

$t = N_s - 2$ 時刻的後向機率 $\beta_{N_s-2}^i(s_{N_s-2})$ 的計算，先檢視此機率的定義

$\beta_{N_s-2}^i(s_{N_s-2}) @ P(\mathbf{Y}_{N_s-1}^{\mathcal{G}} | u_{N_s-2} = i, S_{N_s-2} = s_{N_s-2})$ 與後向遞迴式中所定義之過渡機率 $\gamma_{j,i}(\mathbf{Y}_{N_s-1}^{\mathcal{G}}, s_{N_s-1}, s_{N_s-2})$ ，利用邊緣機率將 $\beta_{N_s-2}^i(s_{N_s-2})$ 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的遞迴關係式 (3.11) 式做比較：

$$\begin{aligned}
& \beta_{N_s-2}^i(s_{N_s-2}) \\
&= \sum_j \sum_{s_{N_s-1}} P(u_{N_s-1} = j, S_{N_s-1} = s_{N_s-1}, \mathbf{Y}_{N_s-1}^{\mathcal{G}} | u_{N_s-1} = j, S_{N_s-2} = s_{N_s-2}) \\
&= \sum_j \sum_{s_{N_s-1}} \gamma_{j,i}(\mathbf{Y}_{N_s-1}^{\mathcal{G}}, s_{N_s-1}, s_{N_s-2}) \\
& @ \sum_j \sum_{s_{N_s-1}} \beta_{N_s-1}^i(s_{N_s-1}) \times \gamma_{j,i}(\mathbf{Y}_{N_s-1}^{\mathcal{G}}, s_{N_s-1}, s_{N_s-2}) \tag{4.19}
\end{aligned}$$

此結果亦顯示 $\beta_{N_s-1}^i(s_{N_s-1}) = 1$ 符合遞迴關係式的機率運算。

此外，過渡機率 $\gamma_{j,i}(\mathbf{Y}_{N_s-1}^{\mathcal{G}}, s_{N_s-1}, s_{N_s-2})$ 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，展開式如下：

$$\begin{aligned}
& \gamma_{j,i}(\mathbf{Y}_{N_s-1}^{\mathcal{G}}, s_{N_s-1}, s_{N_s-2}) \\
&= P(u_{N_s-1} = j, S_{N_s-1} = s_{N_s-1}, \mathbf{Y}_{N_s-1}^{\mathcal{G}} | u_{N_s-2} = i, S_{N_s-2} = s_{N_s-2}) \\
&= P(\mathbf{Y}_{N_s-1}^{\mathcal{G}} | u_{N_s-1} = j, S_{N_s-1} = s_{N_s-1}, u_{N_s-2} = i, S_{N_s-2} = s_{N_s-2}) \\
& \times P(u_{N_s-1} = j | u_{N_s-1} = i, S_{N_s-1} = s_{N_s-1}, S_{N_s-2} = s_{N_s-2}) \\
& \times P(S_{N_s-1} = s_{N_s-1} | S_{N_s-2} = s_{N_s-2}, u_{N_s-1} = i) \tag{4.20}
\end{aligned}$$

其中傳輸訊號 $u_{N_s-1} = 0$ 為額外加入的歸零訊號，屬必然事件，所以 $u_{N_s-1} \neq 0$ 的發生機率將設定為零，同理，其他時刻也都應考量是否為歸零訊號而給予相同的設定。

由於解碼的遞迴演算法建構在迴旋編碼先天基礎上，須在演算過程支出使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，而在系統的模擬實驗中，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 SCCD 的架構中，我們知道若能夠在接收端利用訊號源之間的殘餘冗息將會有效地提升系統的解碼效能。而在第二章節的訊源冗息分析中可以得到，殘餘冗息在符號層級之間存在的資訊將會比位元層級之間還要多。因此，在第三章提出了一個向量量化傳輸系統，並使用基於符號 BCJR 演算法的軟性輸出解碼架構。在本章，我們將這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。

本章的內容將分為兩小節來進行實驗之模擬與結果之分析討論。第一節我們將模擬第 3.1 節所介紹之傳輸系統，在接收端則是採用第 3.3 節中所提出之基於符號軟性輸出通道解碼的系統架構，並探討運用在不同程度的量化符號假設下所存在的殘餘冗息在接收端作解碼工作的效能評比與分析。在第二節中，將此軟性輸出通道解碼應用於歐洲電信標準局 (European Telecommunication Standards Institute, ETSI) 所制訂的分散式語音辨認架構 (Distributed

Speech Recognition, DSR)。在 DSR 標準的接收端所制訂的錯誤隱匿機制(error mitigation scheme)亦有做錯誤回復的解碼工作，其原理將在第二小節中介紹。最後，我們將在解碼端的部分使用我們所提出的軟性輸出通道解碼機制，比較基於位元與基於符號的 BCJR 演算法在運用了位元之間或符號之間所存在的殘餘冗息資訊，再加上最大後驗機率的估測之後，在解碼效能上所反應出來的效果，並且與 DSR 原本所制訂的錯誤隱匿機制作比較。

5.1 基於符號的軟性輸出解碼模擬

5.1.1 系統模擬之步驟說明

在此節之內容中，主要將結合第三章及第四章之演算法進行系統模擬與結果分析。系統中訊號源 v_i 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：

$$v_n = \rho_{vv} v_{n-1} + n_n$$

其中 n_n 表平均值為 0 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 20000 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 LBG 演算法 (Linde-Buze-Gray

algorithm) 所產生出來的向量量化器 (vector quantizer)，其中碼書維度設定為 2，因此將 20000 點訊號源送入此量化器後將可以得到 10000 個索引，各個索引則使用 $M = 6$ 位元表示。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) [15]。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 300 個量化索引與額外附加迫使狀態回歸至零的尾巴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，這裡我們使用的是圖 3.2 編碼率 $r=1/2$ 和限制長度為 $M_c=4$ 的編碼器。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)，即傳送過程中僅受到白色高斯雜訊的通訊干擾。

通道編碼後之位元組合在通過可加性高斯白雜訊通道之後，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格區塊，此處每一個格子架構區塊將解碼出 300 個量化索引以及尾巴位元的軟性輸出訊息。在經由軟性輸出通道解碼之後，我們僅留下 300 個量化索引之軟性輸出訊息，並送入最大後驗機率解碼器作估測得到 \hat{u}_i ，並經由量化碼書查表比對重建出訊號 \hat{v}_i 。

對照第三章以及第四章的介紹，我們的實驗流程圖如圖 5.1，主要探討的是在接收端，當考慮符號 u_t 的位元數 M_d 不同分別去做解碼，系統效能將會有如何的提升。因為在傳送端使用的是 6 位元的量化器，所以我們取其最大公因數分別做實驗，這次的實驗中，我們模擬了四種情況： $M_d = 1$ ， $M_d = 2$ ， $M_d = 3$ ， $M_d = 6$ ，並假設符號與符號前後之間的關聯性可以模擬為一階的馬可夫程序 (AK1)，亦即利用了一階事前消息 $P(u_t | u_{t-1})$ 。其中當 $M_d = 1$ 時，此基於符號的軟性輸出通道解碼退化即為基於位元的軟性輸出通道解碼。在這裡，我們另外產生 100000 個索引，依不同的符號等級以 M_d 個位元分割後做訓練，即可得到一階事前消息 $P(u_t | u_{t-1})$ ，其中 $M_d = 1, 2, 3, 6$ 。再將此事前消息代入前向與後向機率做計算，如此即可求得蘊含了量化後符號間之冗餘訊息的解碼後符號 u_t 的後驗機率，分別以 $(AR, M_d = 1)$ $(AR, M_d = 2)$ $(AR, M_d = 3)$ $(AR, M_d = 6)$ 表示，實驗結果如圖 5.2 所示。

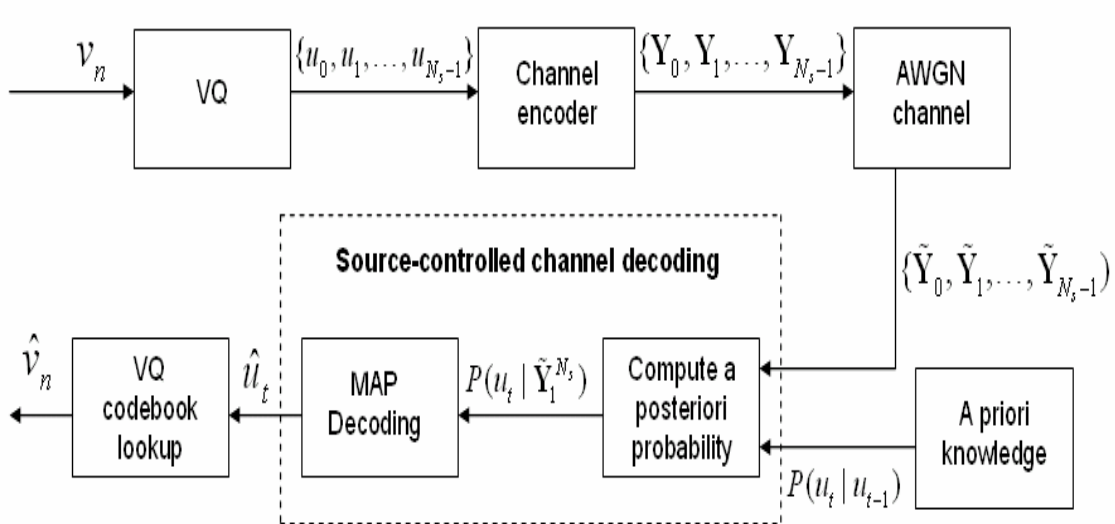


圖 5.1 實驗流程圖

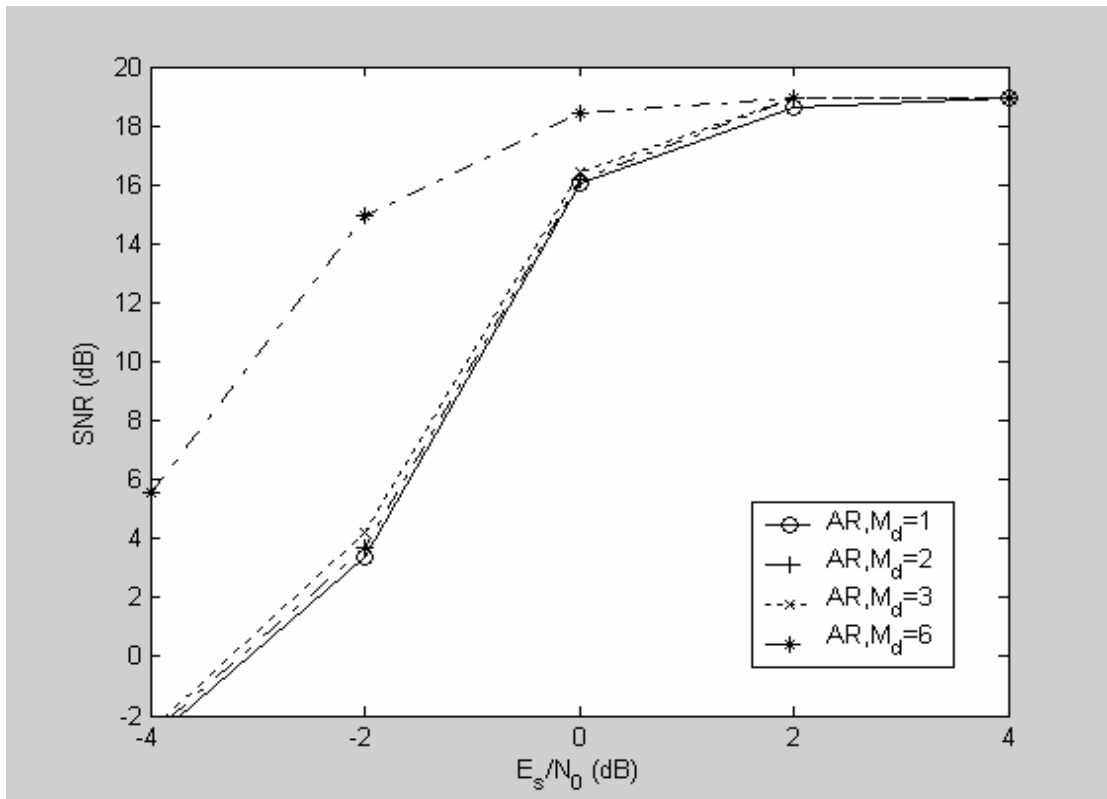


圖 5.2 基於符號的軟性輸出解碼之系統模擬

5.1.2 結果分析

圖 5.2 中，系統之評估為使用訊號源參數 v_n 和重建參數 \hat{v}_n 之間的

訊號雜訊比 (signal-to-noise ratio, SNR) 作為衡量：

$$\text{SNR} = 10 \log_{10} \frac{v_n^2}{(v_n - \hat{v}_n)^2} \quad [\text{dB}]$$

在經由第二章的訊源冗息的分析之後，我們知道當單一符號 u_i 的位元數 M_d 越來越大時，符號與符號間存在的殘餘冗息 ρ_i 越大，這裡皆考慮一階事前消息 $P(u_i | u_{i-1})$ ，也就是我們可以利用的殘餘冗息越多，可以期望的是在接收端對於解碼的幫助會更大，觀察圖 5.2 可以得到相對應於第二章的驗證。在相同的通道環境下，當 M_d 增加， ρ_i 增加，SNR 也就比較高。值得注意的是，在第二章的分析中，可以發現雖然當代表一符號的位元數從 $M_d=1$ 到 $M_d=3$ ，其對應的 ρ_i 有跟著提升，但是提升的幅度不大。一直到 $M_d=6$ 時，也就是符號的位元數等於量化索引的位元數，此時利用的事前資訊是索引之間的殘餘冗息， ρ_k 才有很顯著的增加。因此，我們可以預期的是，在 $M_d=6$ 的情況，SNR 應該也很明顯的提升，從圖 5.2 看來，確實得到我們預期的結果，當通道環境越差，效能的提升更大。順便一提，在圖中當雜訊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，四條曲線被約束在一定的值，效能差不多，此值為量化器所使用的量化位元數所影響，本實驗中，此臨界值為 18.955009 dB。

5.2 分散式語音辨識的應用

5.2.1 通道錯誤緩和機制

近來，在將自動語音辨識技術轉移至行動或是 IP 網路上使用的概念上，DSR 被認為是一個很有效的方法。在 ETSI 的標準 ETSI ES 202 212(v. 1.1.1)裡，描述了整個 DSR 系統中的語音處理、傳輸以及品質效能。標準中並定義了前端語音參數的擷取，以及一個將語音的輸入傳送至伺服器端的語音辨識系統的編碼機制。DSR 的根本概念就是在局部的前端擷取語音參數，再經由資料通道(data channel)傳送至後端計算量較複雜的辨識器。主要是因為若直接將語音傳輸時，往往因為低位元語音編碼率以及通道傳輸錯誤使得系統效能嚴重地下降。所以 DSR 捨棄語音通道(voice channel)，而是代之以有錯誤保護的資料通道傳輸參數，圖 5.2 即為 DSR 的架構圖。因為我們現在想要探討的是，在解碼端利用我們所提出的 SCCD 替換 ETSI 標準原先制定的錯誤緩和(error mitigation)機制之後，在系統效能上的改善，所以下來我們將著墨於伺服器端的討論。

在標準中所訂定之參數擷取演算法將產生一 14 個元素的向量，其中包含了 13 個梅爾倒頻譜係數 (Mel Frequency Cepstrum Coefficients, MFCC)，從 c_0 到 c_{12} ，和 1 個對數能量參數 (log-energy coefficient)。語音信號的分析可被取樣為 8k 赫茲, 11k 赫茲以及 16k 赫茲，每個框架長度 25 微秒且每 10 微秒做一次平移。而這些參數將

被壓縮為一 4800 bps 的資料串以作為傳輸。使用的量化器為分割向量量化器(split vector quantizer, SVQ)，其中 14 個元素將以每兩個參數為一組的方式被分為七個子集合 (c_1 和 c_2 , c_3 和 c_4 , α , c_0 和 log-energy)。每一個子集合利用權重歐式距離(weighted Euclidean distance)分別產生與其相對應的碼書，其中 c_1 到 c_{10} 這五組的碼書以 6 位元做量化， c_{11} 到 c_{12} 這一組以 5 位元做量化， c_0 和 log-energy 則以 8 位元做量化。另外，還有一個語音行動偵測(voice activity detection, VAD)位元會跟隨在 c_{11} 到 c_{12} 這一組的後面，這 44 個位元組合而為一個框架(frame)。

而這些量化後的位元串將被組合至一個 138 個八位元組(octet)的序列稱為多重框架(multiframe)，如表 5.1 所示。在多重框架的格式中，前兩個八位元組為同步(synchronization)序列，接下來的 4 個八位元組為標頭(header)序列，和 138 個八位元組的框架封包串，其中將了 24 個框架每兩兩分為一組。而每一組封包對(frame pair)有 88 個位元，並利用這些位元編碼出 4 位元的循環冗息碼(cyclic redundancy code, CRC)跟隨在這封包對之後如表 5.2。CRC 並沒有錯誤回復的功用，但是最大作用就是在接收端可以做錯誤偵測，此編碼也被廣泛地使用。

另一方面，我們要介紹 ETSI 所制訂的錯誤緩和機制。在接收

端解碼完之後，此錯誤緩和機制有兩種判斷法則：CRC 檢測以及資料一致性。CRC 的檢測是當接收到解碼後的序列之後，我們將利用接受到的框架對做 CRC 的計算，並再與接收序列中對應此框架對的 CRC 加以比對，如果不一樣，那我們就判定此框架對為錯誤。除此之外，資料的一致性則是經由演算法判斷封包對中的封包是否達成最小連續性，如果不是的話，那我們仍然判定此框架對為錯誤。這是一個偵測錯誤叢集(error bursts)的有效方法，假設錯誤叢集發生，連續 $2 \times B$ 個框架被判定為錯誤，那麼前 B 個框架將被這錯誤叢集之前的最後一個正確框架取代，後 B 個框架則被這錯誤叢集之後的第一個正確框架所取代。

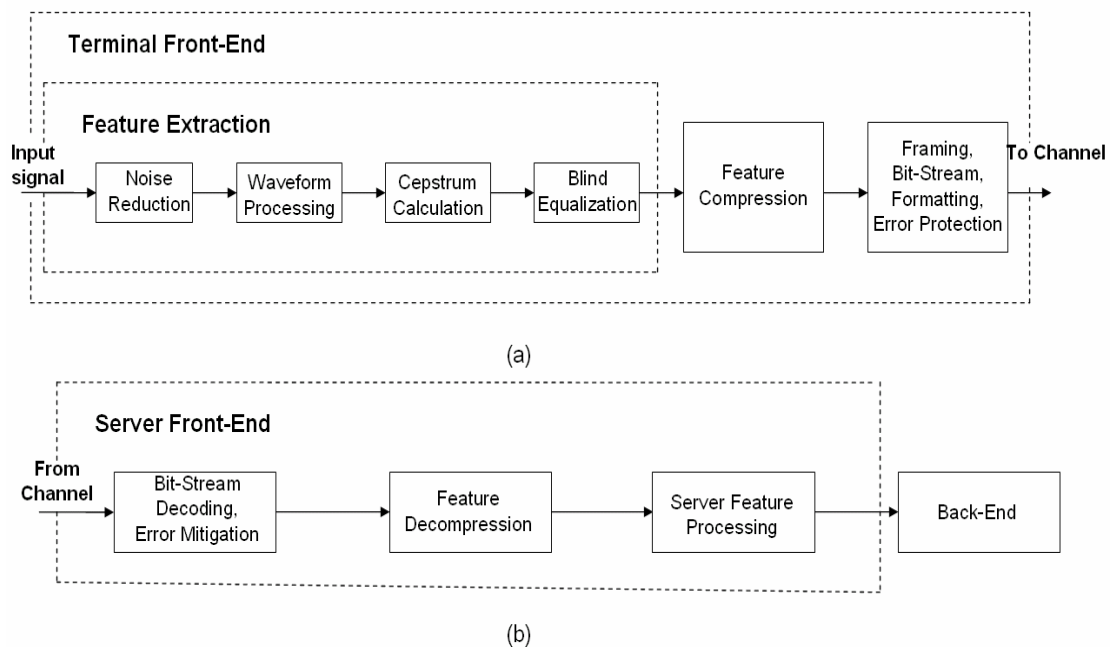
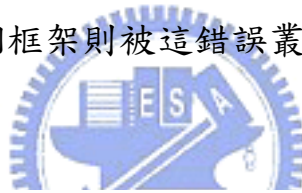


圖 5.3 DSR 格子架構圖 (a)終端機 (b) 伺服器

表 5.1 多重框架的格式

Sync	Header	Frame Packet Stream
2 octets	4 octets	138 octets
← 144 octets →		

表 5.2 受 CRC 保護的封包串

Frame #1	Frame #2	CRC #1-2	...	Frame #23	Frame #24	CRC #23-24
44 bits	44 bits	4 bits		44 bits	44 bits	4 bits
← 138 octets /1104 bits →						

5.2.2 實驗模擬

在本實驗中，我們假設除了量化後的參數以外的資訊都不會發生錯誤，其中包括了同步序列，標頭序列，CRC，以及 VAD 位元，也就是說，我們在接收端先做框架的分割，只考慮將七個參數共 43 位元送至通道中，再接收端再依其原格式重新組合而得原本的形式，並再送至 ETSI 制訂的錯誤緩和機制，或是我們在第三章及第四章所提出的 SCCD 做解碼，流程圖如圖 5.1，再送至語音辨識器。

我們使用不限制字串長度的中文數字串做實驗模擬，使用的中文數字串資料庫事由 50 個男聲以及 50 個女聲錄製而成，每一個說話者錄製十個音檔，而且每一個音檔分別有 1 到 9 不等的數字，其中，90 個發聲者(45 個男聲以及 45 個女聲)所錄製的音檔做為訓練語料，剩下的 10 個發聲者所錄製的做為測試語料，訓練語料與測試語料分別有 6796 以及 642 個數字。在伺服器是由 HTK 軟體做辨識，中文數字

被模擬為隱藏式馬可夫模型 (Hidden Markov Models, HMMs)，每一個字以 8 個狀態模擬其統計模型，而每一個狀態均以 64 個高斯分佈混合模型來近似觀察值的機率分佈。除此之外，並分別以 3 個狀態的 HMMs 以及一個狀態的 HMMs 模擬間歇與停頓，即發音過程前後端與過程中字和字之間過渡時期內訊號的統計模型。在辨識中，12 個梅爾反頻譜係數以及對數能量參數再加上速度 (1 階微分) 與加速度 (2 階微分) 都需要納入考慮。

實驗模擬結果如圖 5.4 所示，在這裡使用的通道環境一樣是可加性高斯白雜訊通道。其中我們模擬了三種情況，第一條曲線 ETSI-MIT 就是在解碼完後利用 ETSI 所制訂的錯誤緩和機制改善因通道所產生的錯誤所造成的影響，SCCD1 則是利用以位元為基礎的 BCJR 演算法做解碼，考慮位元與位元間之間 (即 $M_d=1$) 殘餘的關聯性，SCCD2 則是利用以索引為基礎的 BCJR 演算法，分別以七組參數量化的位元數 ($M_d=6,6,6,6,6,5,8$)，依不同參數的索引層級相關性 $P(u_t|u_{t-1})$ 做解碼。

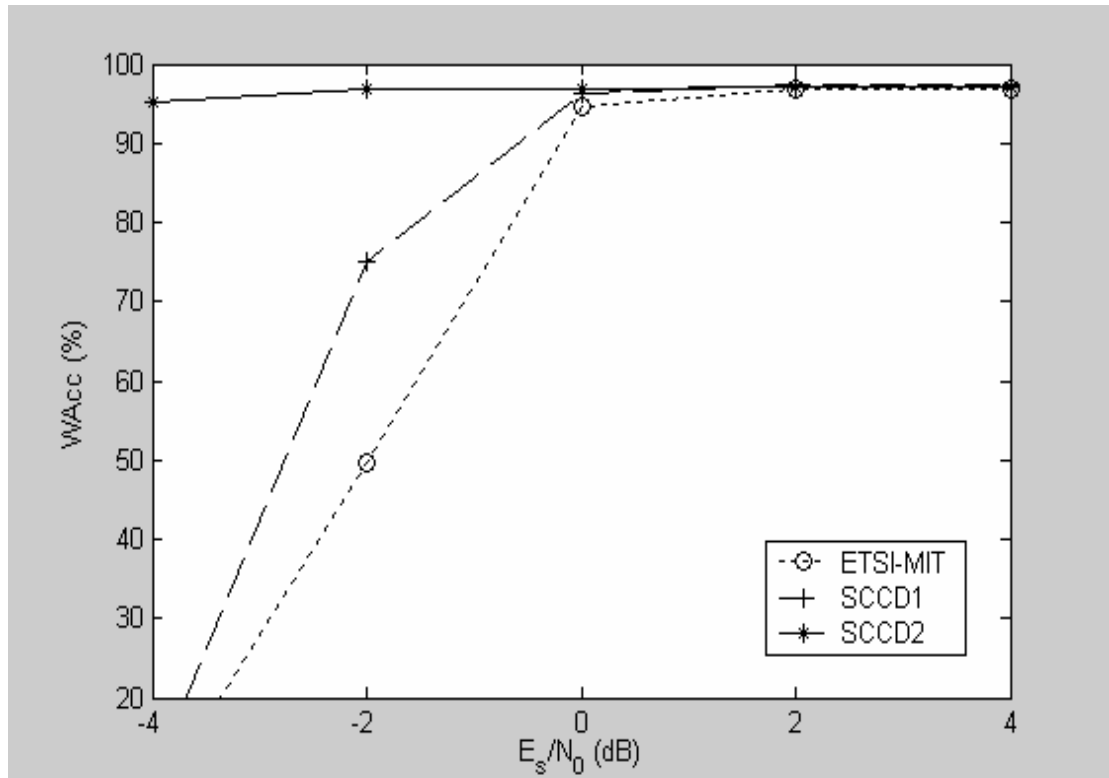


圖 5.4 輸出軟性通道解碼應用於 DSR 的模擬

5.2.3 結果分析

在圖 5.4 中，縱座標為 HTK 軟體中所定義的字元辨識率(Word accuracy, WAcc)。首先，我們在接收端先利用了 ETSI 中的錯誤緩和機制得到了第一條曲線 ETSI-MIT 之後，以這條曲線做為參考基礎，進而在接收端使用了我們所介紹的 SCCD，可以發現，當我們以位元為基礎的 BCJR 演算法 SCCD1 做解碼時，因為利用了位元之間的殘餘冗息，在與 ETSI-MIT 相比之下，將可以使得字元辨識率做微幅的提升。但不幸的是，當 $E_s/N_0 = -4$ dB，因為通道環境太差，所以在解碼端已經沒有辦法作有效地解碼，辨識結果幾乎和 ETSI-MIT 差不

多，約只有 10%~12%，也就是說辨識率相當於我們任意地猜測結果，那麼此系統在這環境下已經沒有參考價值。但是當我們在接收端使用了以索引為基礎的 BCJR 演算法 SCCD2，此時的符號位元數等於在 ETSI 中 VQ 系統的索引位元數 ($M_d = 6, 6, 6, 6, 6, 5, 8$)，所以利用到的是索引與索引之間所殘留的大量殘餘冗息，可以看出辨識率明顯優於前兩條曲線。這不但驗證了我們在第二章的推論，也確定了在第三章 SCCD 的概念中，當通道環境非常差時，SCCD 將更倚重訓練得到的事前訊息甚過於所接受到的序列，使得辨識效能達到極良好的改善。



第六章 結論與未來展望

6.1 結論

在數位通訊系統的傳輸過程中，由於通道環境的嚴重雜訊干擾，導致在接收端經過通道解碼之後，仍無法有效地更正其錯誤。所以陸續有許多合併音源通道解碼的相關研究，也驗證了訊源編碼以及通道編碼應該合併考慮設計，而且不論訊號源的事前資訊或是通道的事前資訊都可以在接收端提供資訊以提升解碼效能，降低位元錯誤率。本

論文提出一個以軟性輸出通道解碼為主的錯誤隱匿機制，主要是利用訊號源在經由訊源編碼後的編碼位元串之間所遺留的殘餘冗息，提升在接收端通道解碼的錯誤更正能力。在此訊源控制通道解碼機制中，使用了 BCJR 演算法，並做最大後驗機率的估測。有別於傳統 BCJR 演算法是以位元為基礎的解碼架構，我們提出以符號為基礎的 BCJR 演算法，利用符號與符號之間的殘餘冗息以提供接收端在解碼工作時更多的資訊。我們首先在第二章分析訊號源在經由訊源編碼器後，將量化索引分割為數個符號後，不同層級的符號與符號之間所殘留的訊源冗息。並在第五章將訊號源模擬為馬可夫程序的實驗模擬中驗證我們的假設，確實隨著符號大小的增加，訊源冗息跟著增加，並帶給接收端更多可以倚靠的資訊，尤其當通道環境劣時，效能提升越大。最後，並將此軟性輸出通道解碼機制應用於 ETSI 制訂的分散式語音辨識系統，並考慮了以位元為基礎和以索引為基礎的 BCJR 演算法，比較不同層級的一階事前消息對於通道解碼能力的幫助，也驗證了當通道的環境越差，從訊源編碼器得到的事前消息越重要，對於解碼的效能提供很大的幫助，也證明了我們所提出的軟性輸出通道解碼確實對於傳輸系統在接收端能夠提供很顯著的改善。

6.2 未來展望

在第五章的實驗模擬，我們證實訊號源在訊源編碼之後所遺留的殘餘冗息結合軟性輸出通道解碼確實大幅提升錯誤更正的能力。但是，我們在模擬中所使用的通道環境僅考慮無記憶性的可加性高斯白通道，而這樣的通道並不符合真實多變的通道環境。因此在未來，可以針對通道模擬環境加以改善，並且考慮記憶性的通道環境，例如 Glibert 通道或是 Fritchman 通道，將通道特性加以考慮相信亦能對解碼效能再多做提升。另外，在以符號為基礎的 BCJR 演算法中，當索引位元數很大，在 BCJR 演算法中的前向機率與後向機率推導以及柵狀圖將更為龐大且複雜，在解碼的效率上將有所延遲。若是在實際應用上將產生不良影響，所以在以符號為基礎的 BCJR 演算法中為了因應不同的訊源編碼器可能輸出較大的索引位元數，勢必得多瞭解演算法的推導並將其簡化的必要。對於此軟性輸出通道解碼所能提供的改善在第五章已經可以知道，若能再對此架構多加以改善，並真實地應用於人們的生活之中，相信將帶給人類更多的便利。



- [1] C.E. Shannon, "A Mathematical Theory of Communication," *Bell Syst. Tech. J.*, vol.27, pp.379-423, pp.623-656, 1948.
- [2] T. Fingscheidt and P. Vary, "Softbit speech decoding: A New Approach to Error Concealment," *IEEE Trans. Speech and Audio Processing*, vol.9, no 3, pp 240-251, March 2001.
- [3] A.M. Peinado, V. Sanchez, J.L. Perez-Cordoba, and A. Torre, "HMM-based channel error mitigation and its application to distributed speech recognition," *Speech Communication*, vol.41, pp. 549-531, 2003.
- [4] H.U. Reinhold, and I. Valentin, "Soft features for improved distributed speech recognition over wireless networks," *8th International Conference on spoken Language Processing*, pp.2125-2128, Jeju Island, Korea, Oct 2004.

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

[6]ETSI ES 202 212 v1.1.1. Digital speech recognition; extended advanced front-end feature extraction algorithm; compression algorithms; back-end speech reconstruction algorithm. November 2003.

[7]S. Lin and D. J. Costello, Jr., *Error Control Coding*. Englewood Cliffs, NJ: Prentice-Hall, 2004.

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

[9]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.

[10]L.T. 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.

[11]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.

[12]N. Phamdo, and N. Farvadin, "Optimal detection of discrete Markov source over discrete memoryless channel-application to combined source-channel coding", *IEEE Trans. Inform. Theory*, vol.40, pp.186-193, Jan.1994.

[13]V. Cuperman, F.H. Liu, and P. Ho, "Robust vector quantization for noisy channel using soft decision and sequential decoding," *Eur. Trans.*

Telecomm., vol.5, no.5, pp.7-18, Sept 1994.

[14]K.P. Ho, “Soft-decoding vector quantizer using reliability information from turbo-codes,” *IEEE Commun, Lett.*, vol.3, pp.208-210, July 1999.

[15]N.S. Jayant and P. Noll, *Digital Coding of Waveforms*. Englewood Cliffs, NJ:Prentice-Hall, 1984.

