行政院國家科學委員會專題研究計畫成果報告
具錯誤更正能力的持續持續長度限制碼 (II)

Run Length Limited Codes with Error Correcting Capabilities(II)

計畫編號：NSC 89-2213-E-002-121
執行期間：89 **年** 8 **月** 1 **日至** 90 **年** 7 **月** 31 **日**

主持人：林茂昭 國立臺灣大學電信所

一、中文摘要

在某些數位傳輸通道中，如數位磁性記錄器，或是數位光記錄器等，為了避免位元信號之間的相互干擾，以及為了考量時序同步信號的產生，其位元序列必須受到某些持續長度之限制。以”+1”**及**”-1”**為位元之持續長度限制序列可轉換為以**”1”**及**”0”**為位元之持續長度限制序列，稱之為**(d,k)**序列，其中** d **為**”0”**位元之最小持續長度，**k **為**”0”**位元之最大持續長度。由許多（**d,k**）序列所成之集合為**(d,k)**碼，或稱持續長度限制碼**(run-length-limited code, RLL code)。

在國科會計畫 NSC81-0404-002- E -002-002 **中我們曾經設計出一些具錯誤更正能力之**(d,k)**碼。我們發現這些碼在碼率， 錯誤更正能力及複雜度方面仍然有許多改善餘地。本計畫中以多層次編碼方式設計出具有低複雜度，高錯誤更正能力及高碼率之持續長度限制碼。本計畫也有一個很好的附帶結果，即是找到一個設計低複雜度二元**(n,n-1)**迴旋碼之方法。**

英文摘要

For some digital communication channels, such as the digital magnetic recorder, or the digital optical recoder, to alleviate the problem of inter-symbol interference and to assist the synchronization, the associated data sequences must be subjected to some run length constraint. A run length limited sequences with symbols of “+1” and “-1” can be convert to a sequences with symbols of “1” and “0”, which is called a (d,k) sequence, where d is the minimal run length of 0’s between two consecutive 1’s and k is the maximal run length of of 0’s between two consecutive 1’s. A (d,k) code is a collection of some (d,k) sequences. Such code is also called run-length-limited (RLL) code.

In the project NSC81-0404-E-002-002, we have designed some (d,k) codes with error correcting capabilities. Recently, we find that there are new techniques which can be used to further improve the error-capability, coding rates and decoding complexity of (d,k) codes. Based on the concept of multilevel coding, in this project, we propose several classes of powerful (d,k) codes with high error correcting capabilities, high coding rates and low decoding complexity. There is a by product of this project. That is we find a method to design (n,n-1) convolutional codes with low trellis complexity.

二 、計畫的緣由與目的(Goals)

In magnetic recording systems, run length constraints on data are usually required to reduce the effect of intersymbol interference and to support bit synchronization. A class of constrained data sequences is called (d,k) sequence, where d is the minimum run length of 0’s between 1’s and k is the maximum run length of 0’s between 1’s. The set of all binary (d,k) sequences is denoted by $C^n(d,k)$. The set of all concatenatable (d,k) sequences of lengh n which has the largest cardinality is denoted by $C^n_{\max}(d,k)$. is used to denote the set of all binary (d,k)sequence of length n. Based on the set $C^n_{\max}(d,k)$, a class of (d,k) trellis codes with single-error-correcting capability was proposed in [1]. This code has the drawback of low error-correcting capability and high decoding complexity. In [2], the concept of Steiner triple system was used to improve the error-correcting capability of [1]. The penalty is the decreased coding rate. In [3], a class of (d,k) trellis codes is proposed, which is based on multilevel coding technique used in [4] and [5]. The trellis code proposed in [3] is much better than that proposed in either [1] or [2]. However, there is still room for improvement.

In this project, we use several techniques

to modify the trellis codes of [3] to find some good (d,k) trellis codes. For the first one, we use a convolutional processor [6] to replace the delay processor used in [3]. For the second one, we propose a method to find better partition rules for $C^n_{\max}(d,k)$. For the third technique, we add two bits to each sequence in $C^n(d,k)$ Examples are provided to demonstrate the superiority of the (d,k) codes designed by the proposed techniques. In the search for good (d,k) trellis codes, we also find a method that can design good (n,n-1) convolational codes with low trellis complexity. The basic idea is that we find an optimum permutation for any given (n,n-1) binary convolutional code that will yield an equivalent code with the lowest state complexity.

三、研究方法與成果 (Methods and Results)

In [1], a class of RLL trellis (Lee-Wolf code) which have single error correcting capability was proposed. A Lee-Wolf code is constructed by partitioning $C^n_{\max}(d,k)$ into two subsets, C_e and C_o, and using a rate (m-1)/m convolutional code with 2^{m-1} states to select RLL n-tuples in C_e and C_o, where C_e and C_o are the sets of even weight and odd weight n-tuples in $C^n_{\max}(d,k)$ respectively. It is required that $|C_e| \geqq 2^{m-1}$ and $|C_o| \geqq 2^{m-1}$.

In [5], a multilevel coding technique is used to construct trellis coded modulation systems. Consider a signal set Ù that consists of 2^m signal points. Each signal point in Ù is marked by ù(ß), where ß =($s_1,s_2,\ldots,s_m$),$s_i \in \{0,1\}$, i=1,2,…,m. The encoding of the system in [5] is illustrated in Fig.1. At time t, an r-bit message ß(t) is encoded by a rate r/m convolutional code C to result in an m-bit code branch $\tilde{\nu}(t) = (\nu_1(t), \nu_2(t), ..., \nu_m(t))$. The output of the multilevel delay processor is $\tilde{s}(t) = (s_1(t), s_2(t), ..., s_m(t))$. The output of the signal mapper is $\mu(\tilde{s}(t)) = z(t) \in \Omega$, where $s_1(t) = \nu_1(t-(m-l)^a),\ for\ l = 1,...,m$ and ë is a constant. The resultant trellis code is denoted by T. If the mapping between the m-tuple ß and the signal point ù(ß) is appropriately designed, a large free distance for T can be achieved for a signal set Ù such as 8PSK or$\{0,1\}^4$. Let Ä(z_1,z_2) denote the distance measure between z_1 and z_2, where z_1, $z_2 \in$ Ù. If Ù is a collection of binary n-tuples for n ≥ m, then Ä(z_1,z_2) is the Hamming distance between z_1 and z_2. if Ù is a collection of binary n-tuples for n ≥ m. Set partition for Ù is needed. Let Ù = $Ù_0/Ù_1/\ldots/Ù_m$ denote the partition chain for Ù provided that Ù is a linear space. Each partition $Ù_{j-1}/Ù_j$ results in two cosets of $Ù_j$. We define

$$\Delta_j = \begin{cases} \min\limits_{s_j \neq s_{j'}} \{\Delta(\mu(\hat{s}),\mu(\hat{s}')) : \mu(\hat{s}),\mu(\hat{s}') \in \Omega \ , j=1\} \\ \min\limits_{s_j \neq s_{j'}} \{\Delta(\mu(\hat{s}),\mu(\hat{s}')) : \mu(\hat{s}),\mu(\hat{s}')\} \in \Omega, \\ s_i = s_i' \quad for: 1 \leq i \leq j, 1 < j \leq m \end{cases} \tag{1}$$

We then have an m-level distance profile {△1,…,△m}. The free distance of T is lower bounded [3,4] by

$$\Delta_{LB}(^a) = \min_{\substack{\hat{v} \in C \\ \hat{v}(0) \neq 0}} \sum_{t=0}^{a-1} \sum_{j=1}^{m} \nu_j(t) \Delta_j \tag{2}$$

The bound △LB(ë) is an increasing function of ë and will become a constant value △free when ë exceeds a threshold number $ë_{min}$. A suboptimal decoding for T can be implemented by using the trellis of C [4]. The suboptimal decoding can fully utilize the error-correcting capability guaranteed by △free. Hence, the decoding complexity is only slightly higher than that of C.

The coding scheme just described is used in [3] to construct RLL codes with good correcting capabilities. It difficult to find a general rule for choosing Ù and ù that can yield a trellis code T with the largest free distance for a given (d,k) constraint and a arbitrary n. In [3], Ù and ù are chosen by following the partition used in [1]. That is, we choose Ù=$Ù_{s1=0}$ ∪ $Ù_{s1=1}$, where $\Omega_{s1=0} = \{\mu(s_1=0, s_2, ..., s_m) \mid s_2, ..., s_m \in (0,1)\} \in C_e$ and $\Omega_{s1=1} = \{\mu(s_1=0, s_2, ..., s_m) \mid s_2, ..., s_m \in (0,1)\} \in C_o$ .Note that Ù used here is not a linear space. However, we can design the partition in a similar way except that the partition

$Ù_{j-1}/Ù_j$ here results in two subsets which may not be cosets of $Ù_j$. Now, we have a distance profile of$\{\triangle_1=1,\triangle_2=2\ldots,\triangle_m \geqq 2\}$. Using the distance profile, RLL codes [3] are found which are listed in table 1.

Scheme 1:Convolutional processor

In [6], a trellis coding scheme better than that proposed in [3] is proposed. The encoder is shown in Fig. 2 which is implemented by replacing the delay processor in the encoder of Fig. 1 by a convolutional processor. The convolutional processor is a rate 1 convolutional code with generator matrix P. Here, we denote the resultant trellis code by T'. Let $a_1=0, a_2,\ldots, a_m$ be nonnegative integers such that $a_j \le \Delta_j/\Delta_{j-1}$ for $2 \le j \le m$. We will construct a convolutional processor such that the bit $v_j(t)$ will affect the encoding of a_{j+1} output symbols. Define $\hat{a}_m = 0$ and $\hat{a}_j = \sum_{i=j+1}^{m}(a_i + ¶_i + 1)$ for $1 \le j \le m-1$. Then we have

$$P_{i,i}=D^{\hat{a}i ë},$$

for i=1,2,…,m, and

$$P_{i,i-1} = \begin{cases} 0 & , if \quad a_i = 0 \\ (D^{(i-1)a} + \ldots + D^{(i-1-a_i)a}), & if \quad a_i > 0 \end{cases} \qquad (3)$$

for i=2,3,…,m.

With such a convolutional processor we have

$$s_m(t) = v_m(t), \quad for \quad j = 1,2,\ldots,m-1$$

then we have

$$s_j(t)= \begin{cases} v_j(t-\ ^{a}) & , a_{j+1}=0 \\ v_j(t-\ ^{a}) \oplus v_{j+1}(t-(\ -1)^{a}) \oplus \ldots \oplus v_{j+1}(t-\ +a_{j+1})^{a}), & a_{j+1}>0 \end{cases} \qquad (4)$$

The free distance of T' is bounded [6] by

$$\Delta_{LB}(^{a}) = \min_{\substack{\hat{v}\in C \\ \hat{v}(0)\neq 0}} \sum_{t=0}^{a-1} \sum_{j=1}^{m} v_j(t)(\Delta_j + a_j \Delta_{j-1}) \qquad (5)$$

Similar to the case of trellis code T, the bound $\triangle_{LB}(ë)$ is an increasing function of ë and will become a constant value $\triangle_{free}$ when ë exceeds a threshold number $ë_{min}$. A suboptimal decoding for T' can also be implemented by using the trellis of C [2]. The suboptimal decoding can fully utilize the error-correcting capability guaranteed by $\triangle_{free}$. Hence, the complexity of decoding T' is slightly higher than that of T.

In this report, we use two different generator matrice P and apply the concept from [4] to compute their free distances.

Case 1:

We use $a_1 = 0, a_2 = 1, a_3 = 0$ and $\hat{a}_1 = 4$, $\hat{a}_2 = 2$, $\hat{a}_3 = 0$. Then, we have

$$P=\begin{pmatrix} D^{4a} & 0 & 0 \\ D^{3a} & D^{a} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Case 2:

We use $a_1= 0, a_2= 1, a_3= 1, a_4= 0$ and $\hat{a}_1 = 8$, $\hat{a}_2 = 5$, $\hat{a}_3 = 1$, $\hat{a}_4 = 0$. Then, we have

$$P=\begin{pmatrix} D^{8a} & 0 & 0 & 0 \\ D^{7a} & D^{5a} & 0 & 0 \\ 0 & D^{4a} & D^{a} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The parameters of some of the resultant trellis RLL codes are listed in table 2. We see that there is only slight improvement of no improvement for the codes in table 2 as compared to the codes in table 1.

Scheme 2 : Modified partition:

In [1] and [3], the labeling for the signal set Ù is chosen in a random way except that $s_1=0$ for the n-tuples in the subset of even-weight n-tuples, C_e, and $s_1 = 1$, for the subset of odd-weight n-tuples, C_o.

Now we propose a method that can label the n-tuples in a systematic way. The method of labeling is described as follows.

1. Partition the signal set $C^n_{\max}(d,k)$ into two disjoint subsets(even parity or odd parity).
2. An n-tuple $(c_1,c_2,\ldots,c_n)$ labeled by an index $I = \sum_{i=1}^{n} c_i 2^{i-1} + 1$.
3. Then, we can arrange the n-tuples of $C^n_{\max}(d,k)$ in order according to I. We choose the n-tuples with the largest 2^{m-2} indices and the n-tuples with the smallest 2^{m-2} indices from C_e to form $\Omega_{s1=0} = \{\mu(s_1=0,s_2,\ldots,s_m) \mid s_2,\ldots,s_m \in (0,1)\}$
   We also choose the n-tuples with the largest 2^{m-2} indices and the n-tuples with the smallest 2^{m-2} indices from C_o to form $\Omega_{s1=1} = \{\mu(s_1=0,s_2,\ldots,s_m) \mid s_2,\ldots,s_m \in (0,1)\}$
4. We number these 2^{m-1} n-tuples in C_e by $I_e=1,2,\ldots,2^{m-1}$ consider two n-tuples

$\overline{C}$ *and* $\overline{C}$' with indices I and I' respectively. If I>I' then I_e>I_e'. Similarly, we number the 2^{m-1} n-tuples in C_o by I_o=1,2,…,2^{m-1}.

5. The labeling of s_2,…,s_m for n-tuples in $Ù_{s1=0}$ is given by $S = \sum_{i=2}^{m} s_i * 2^{i-2} + 1$,where we set

$$\begin{cases} S = 2i-1 \ \ ,if\ I_e = i\ for\ odd\ i \\ S = 2i, if\ I_e = i + 2^{m-2}\ for\ even\ i \end{cases}$$

The labeling of s_2,…,s_m for n-tuples in $Ù_{s1=1}$ can be similarly derived from I_o.

Using this new partition (labeling), we rederive the distance profile and the free distances of some trellis codes T and T' which are listed in table 3 and table 4 respectively. We can see that the trellis codes T' using the modified partition have much improved free distances.

Scheme 3 : Inserting 2 bits as buffer

In[7], Immink proposed the method which inserts additional â bits between adjacent (d,k) sequences of length n as merging bits so that the concatenated sequence can still preserve the d and k constraints. From a similar concept we add two bits to the end of each n-tuple in C^n(d,k). If the n-tuple has even parity, then we add two bits 10. If the n-tuple has odd parity, then we add two bits 01. To ensure that the modified (n+2)–tuple can be cascaded without violating the d and k constraints, we must modify the constraints on the head and tail in the n-tuple. Hence we can not use Lee-Wolf's design. The new constraints are given as follows.

1. The tail of the codeword must end with a run of at least d, and at most k-1 consecutive zeros.
2. The head of the codeword must begin with a run of at least d, and at most k-1 consecutive zeros.

With the insertion of two bits, we can find that the minimum Hamming distance between an (n+2)-tuple in $Ù_{s1=0}$ and an (n+2)-tuple in $Ù_{s1=1}$ is 3. Hence the distance profile is improved. We apply this technique of bit insertion to either T or T'. We can see the free distance are increased. The results are listed in table 5 and table 6 respectively.

Good (n,n-1) Convolutional Codes:

In the code search for good (d,k) trellis codes, we usually need good (n,n-1) convolutional codes. We find that it is possible to find an optimum permutation for any given (n,n-1) binary convolutional code that will yield an equivalent code with the lowest state complexity. With this permutation, we are able to find many (n,n-1) binary convolutional codes which are better than punctured convolutional codes of the same code rate and memory size by either lower decoding complexity or better weight spectra.

四、結論與討論 (Concluding Remarks)

In this project, we propose three techniques to construct good (d,k) codes. We compare the trellis codes using the proposed techniques with a class of previously known good (d,k) codes shown in [3]. The first technique can help construct (d,k) codes which achieve slightly increased free distances without sacrificing the coding rates while the penalty is the slightly increased decoding complexity. The second technique is a modified partition rule which can help to achieve large free distances in case that this technique is incorporated into the first technique. The third technique can achieve very large free distance. However, the penalty is the decreased coding rate.

五、參考資料(References)

[1] LEE, P., and WOLF, J.K.: "A general error correcting code construction for run length limited binary channels," IEEE Trans. Inform. Theory, 1989, 35, (6), p.1330-1335

[2]Mao-Chao Lin Technical report, NSC81-0404-E-002-002

[3]H.H. Tang and M.C. Lin, "Class of Multilevel run length limited trellis codes, " Electronics Letters , Volume: 35 Issue: 25,9 Dec 1999Page(s): 2192 –2193

[4]HELLSTERN. G: "Coded modulation with feedback decoding trellis codes," IEEE Conf. Comm., 1993, pp. 1071-1075

[5] J.Y. Wang and M.C. Lin, "On constructing trellis codes with large free distances and low decoding complexities," IEEE Trans. Commun., vol.45, no.9, pp.1017-1020, Sept. 1997.

[6]M.C. Lin; Y.L. Ueng; J.Y. Wang," Two trellis coding schemes for large free distances," Communications, IEEE Transactions on, V(48) 2000 pp. 1286 –1296

[7]Immink, K.A.S. "Run length-limited sequences", Video, Audio and Data Recording, 1990. Eighth International Conference on, 1990 pp. 176 -182

六、圖表 (Figures and Tables)

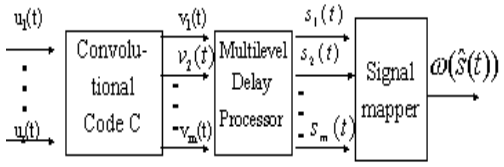

Fig.1 The encoder of T which uses a delay processor and a signal mapper.

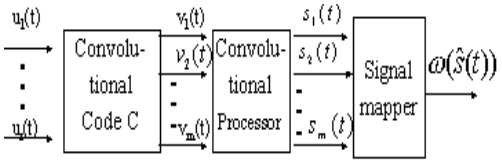

Fig.2 The encoder of T' which uses a convolutional processor and a signal mapper

v	λ_{min}	Generator	(1,3) n/rate d_{free} dist_profile	(2,7) n/rate d_{free} dist_profile	(1,7) n/rate d_{free} dist_profile
2	5	$\begin{pmatrix}1&1&1\\5&7&0\end{pmatrix}$	9/0.222 d_{free}=5 (1,2,2)	10/0.2 d_{free}=5 (1,2,2)	6/0.333 d_{free}=5 (1,2,2)
2	6	$\begin{pmatrix}1&1&1&1\\3&0&1&2\\2&1&3&2\end{pmatrix}$	10/0.3 d_{free}=5 (1,2,2,2)	11/0.273 d_{free}=5 (1,2,2,2)	8/0.375 d_{free}=5 (1,2,2,2)

Table 1: Trellis code T using the original partition

v	λ_{min}	Generator	(1,3) n/rate d_{free} dist_profile	(2,7) n/rate d_{free} dist_profile	(1,7) n/rate d_{free} dist_profile
2	5	$\begin{pmatrix}1&1&1\\5&7&0\end{pmatrix}$	9/0.222 d_{free}=5 (1,2,2)	10/0.2 d_{free}=5 (1,2,2)	6/0.333 d_{free}=5 (1,2,2)
2	6	$\begin{pmatrix}1&1&1&1\\3&0&1&2\\2&1&3&2\end{pmatrix}$	10/0.3 d_{free}=6 (1,2,2,2)	11/0.273 d_{free}=6 (1,2,2,2)	8/0.375 d_{free}=6 (1,2,2,2)

Table 2: Trellis code T' using the original partition

V	λ_{min}	Generator	(1,3) n/rate d_{free} dist_profile	(2,7) n/rate d_{free} dist_profile	(1,7) n/rate d_{free} dist_profile
2	5	$\begin{pmatrix}1&1&1\\5&7&0\end{pmatrix}$	9/0.222 d_{free}=6 (1,2,2)	10/0.2 d_{free}=6 (1,2,2)	6/0.333 d_{free}=5 (1,2,2)
2	6	$\begin{pmatrix}1&1&1&1\\3&0&1&2\\2&1&3&2\end{pmatrix}$	10/0.3 d_{free}=5 (1,2,2,2)	11/0.273 d_{free}=5 (1,2,2,2)	8/0.375 d_{free}=5 (1,2,2,2)

Table 3: Trellis code T using the modified partition

v	λ_{min}	Generator	(1,3) n/rate d_{free} dist_profile	(2,7) n/rate d_{free} dist_profile	(1,7) n/rate d_{free} dist_profile
2	5	$\begin{pmatrix}1&1&1\\5&7&0\end{pmatrix}$	9/0.222 d_{free}=7 (1,2,2)	10/0.2 d_{free}=7 (1,2,2)	6/0.333 d_{free}=5 (1,2,2)
2	6	$\begin{pmatrix}1&1&1&1\\3&0&1&2\\2&1&3&2\end{pmatrix}$	10/0.3 d_{free}=7 (1,2,2,2)	11/0.273 d_{free}=6 (1,2,2,2)	8/0.375 d_{free}=6 (1,2,2,2)

Table 4: Trellis code T' using the modified partition

v	λ_{min}	Generator	(1,3) n/rate d_{free} dist_profile	(2,7) n/rate d_{free} dist_profile	(1,7) n/rate d_{free} dist_profile
2	5	$\begin{pmatrix}1&1&1\\5&7&0\end{pmatrix}$	12/0.167 d_{free}=9 (3,2,6)	12/1.67 d_{free}=6 (3,2,2)	9/0.222 d_{free}=6 (3,2,2)
2	6	$\begin{pmatrix}1&1&1&1\\3&0&1&2\\2&1&3&2\end{pmatrix}$	13/0.231 d_{free}=6 (3,2,2,4)	15/0.2 d_{free}=6 (3,2,2,2)	10/0.3 d_{free}=6 (3,2,2,2)

Table 5: Trellis code T with the insertion of 2 bits

v	λ_{min}	Generator	(1,3) n/rate d_{free} dist_profile	(2,7) n/rate d_{free} dist_profile	(1,7) n/rate d_{free} dist_profile
2	5	$\begin{pmatrix}1&1&1\\5&7&0\end{pmatrix}$	12/0.167 d_{free}=14 (3,2,6)	12/1.67 d_{free}=9 (3,2,2)	9/0.222 d_{free}=9 (3,2,2)
2	6	$\begin{pmatrix}1&1&1&1\\3&0&1&2\\2&1&3&2\end{pmatrix}$	13/0.231 d_{free}=12 (3,2,2,4)	15/0.2 d_{free}=10 (3,2,2,2)	10/0.3 d_{free}=10 (3,2,2,2)

Table 6: Trellis code T' with the insertion of 2 bits