A New Construction Scheme of Convolutional Codes

Wanquan Peng[1] and Chengchang Zhang[2]
[1] Chongqing Vocational Institute of Engineering, Chongqing 400037, China
[2] College of Communication Engineering of Chongqing University, Chongqing 400044,China
Email: 408502@163.com; zcc@cqu.edu.cn

Abstract—A new class of (2*k*, *k*, 1) convolutional codes is proposed based on the method that creating long codes by short ones in this paper, by embedding (2*k*, *k*) double loop cyclic codes in (2, 1, 1) convolutional codes. The structural mechanism of the codes is revealed by defining a state transition matrix and using algebraic method. It is then shown that the code structure is excellent in both proportionality and diversity, so a superior code is easy to be obtained. Simulation results show that, the new convolutional codes present advantages over the traditional (2, 1, *l*) codes in the error-correcting capability and decoding speed.

Index Terms—convolutional codes, state transition matrix, magic square, viterbi decoding

I. Introduction

Convolutional codes are basic Error Correcting Codes (ECC) with memory and good error-correcting capability. Early convolutional codes mainly include the orthogonal convolutional codes [1] which are suitable for majority-logical decoding, the nonsystematic convolutional codes-Quick-look-in codes [2] with a "quick-look-in" feature, and the complementary convolutional codes [3] constructed by two complementary sub-generators. Since the 1980s, Punctured Convolutional Codes (PCC), Tail Biting Convolutional Codes (TBCC) and the Trellis Coded Modulation (TCM) have been widely used [4]-[6]. In the 1990s, a recursive convolutional codes-Recursive Systematic Convolutional Codes (RSC) appeared following the Turbo codes [7]. In recent years, some scholars began to research the convolutional LDPC codes [8]-[11]. This class of convolutional codes can realize iterative decoding based on Belief Proporgation (BP), and has excellent error-correcting performance. To obtain a long free distance, not only a good generated matrix should be depended on, but also the memory length *kl* of (*n*, *k*, *l*) convolutional codes should be increased, which can be achieved by the coding constraint degree *l* or the block length *k*. The orthogonal convolutional codes and the punctured convolutional codes have been successfully used to increase the block length *k*. But *k* is increased quite a little, what's more, the code rate is increased and a bad distance property is caused, which leads to more research on how to increase the coding constraint degree *l*. With the development of computer technology, enumeration type search can obtain the high quality convolutional codes [12] much easier than search by algebra theory. Unfortunately, people are more willing to pay attention to the change of coding structure by only increasing the constraint degree *l* when making the tentative search.

It is indicated that the traditional (2, 1, *l*) convolutional codes are long codes constructed by the (2, 1) even-parity codes in [13], also the essence of the information sharing among blocks in different time is clarified deeply. In this paper, by replacing the (2, 1) even-parity codes with the (2*k*, *k*) linear block codes, and combining with the encoder structure of (2, 1, 1) convolutional codes, a new (2*k*, *k*, 1) convolutional coding scheme, called *first-order magic square convolutional codes*, is presented.

The structure of the paper is as follows: Based on the modificatory encoder given in the (2, 1, *l*) convolutional codes, Section II describes the encoding process of (2*k*, *k*, 1) convolutional codes. In Section III, by using a state transition matrix, the code structure mechanism and the distance property are made a detailed algebraic analysis. In Section IV, we complete the simulation for the soft-decision Viterbi matrix decoding, and make a comparison with the (2, 1, l) convolutional codes in both decoding speed and error-correcting performance. At last, we summarize our discovery finding in Section V.

II. Coding of (2*K*, *K*, 1) Convolutional Codes

The operations following in the paper are calculated according to Galois Field GF(2). A modificatory encoder of (2, 1, *l*) convolutional codes with the embedded zero module is proposed in [13], which can show the mechanism that creating long codes by short ones. When $\boldsymbol{G}=[1\ 1]^{\mathrm{T}}$, $\boldsymbol{M}(t)=[m_0(t)]$, and the generator polynomial matrix of the (2, 1, 1) convolutional codes is:

$$\boldsymbol{G}(\boldsymbol{D})=\begin{bmatrix} g_1 \\ g_2 \end{bmatrix}=\begin{bmatrix} 1+\boldsymbol{D} \\ \boldsymbol{D} \end{bmatrix} \tag{1}$$

We can obtain this encoder, as shown in Fig. 1, where $\boldsymbol{D}$ means the time delay. Unlike the conventional encoder, the input of embedded zero module is g_1+g_2, and not g_2, and the output can be obtained as follows:

Manuscript received Month Day, 2013; revised Month Day, 2013, accepted Month Day, 2013.
Corresponding author email: 408502@163.com.
This work was supported by Natural Science Foundation Project of CQ CSTC under No.cstc2013jcyjA40055, and Foundation of Chongqing Education Commission under No.KJ122003.
doi:10.12720/jcm.8.7.414-420

$$\boldsymbol{C}^{ij}=\boldsymbol{C}_1^{ij}+\boldsymbol{C}_2^{ij}=\begin{bmatrix}m_0(t)+m_0(t-1)\\ m_0(t)+m_0(t-1)\end{bmatrix}+\begin{bmatrix}0\\ m_0(t)\end{bmatrix}$$
$$=\begin{bmatrix}m_0(t)+m_0(t-1)\\ m_0(t-1)\end{bmatrix} \quad (2)$$

We find that (2) is in accordance with (1). In order to construct (2*k*, *k*, 1) convolutional codes, we set afresh that $\boldsymbol{M}(t)=[m_0(t)\ m_1(t)\ m_2(t)\ \ldots m_{k-1}(t)]^{\mathrm{T}}$ is the *k*-bit binary information vector, the $\boldsymbol{D}$ that is called *vector register* delays the *k*-bits information simultaneously, by which way we can obtain $\boldsymbol{M}(t-1)=[\ m_0(t-1)\ m_1(t-1)\ m_2(t-1)\ \ldots m_{k-1}(t-1)\ ]^{\mathrm{T}}$, and the generator matrix of the (2*k*, *k*) linear block codes is:

$$\boldsymbol{G}=\begin{bmatrix}\boldsymbol{I}\\ \boldsymbol{P}\end{bmatrix} \quad (3)$$

where $\boldsymbol{I}$ is the identity matrix, both $\boldsymbol{I}$ and $\boldsymbol{P}$ are $k\times k$ matrix, coding base on (3), we can obtain first:

$$\boldsymbol{C}_1^{ij}=\boldsymbol{G}\times\big(\boldsymbol{M}(t)+\boldsymbol{M}(t-1)\big)=\begin{bmatrix}\boldsymbol{M}(t)+\boldsymbol{M}(t-1)\\ \mathrm{P}\big(\boldsymbol{M}(t)+\boldsymbol{M}(t-1)\big)\end{bmatrix} \quad (4)$$

By this process, we find that the adjoining information vector $\boldsymbol{M}(t)$ and $\boldsymbol{M}(t\text{-}1)$ implement encoding together, thereby the restriction and memorability are realized. Next, in embedded zero module, *k* zeroes are embedded and we obtain the $2k\times 1$ output vector

$$\boldsymbol{C}_2^{ij}=\begin{bmatrix}0\\ \boldsymbol{M}(t)\end{bmatrix} \quad (5)$$

Adding it with (4) the encoding output can be obtained as follows

$$\boldsymbol{C}^{ij}=\boldsymbol{C}_1^{ij}+\boldsymbol{C}_2^{ij}=\begin{bmatrix}\boldsymbol{M}(t)+\boldsymbol{M}(t-1)\\ \mathrm{P}\big(\boldsymbol{M}(t)+\boldsymbol{M}(t-1)\big)+\boldsymbol{M}(t)\end{bmatrix} \quad (6)$$

where superscript *i* and *j* are respectively the decimal number of $\boldsymbol{M}(t)$ and $\boldsymbol{M}(t-1)$, *i*, *j*=0, 1, 2, …, 2^k-1, which can show conveniently the corresponding relation between encoding output and information vector. In the process above, the (2*k*, *k*) linear block codes (hereinafter, "*embedded codes*") are embedded in the (2, 1, 1) convolutional codes, and the two kinds codes are integrated organically, which shows fully the trait that construct long codes with short ones.

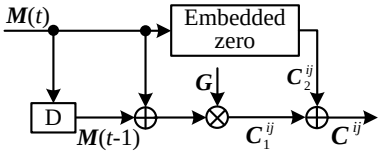


Figure1. (2*k*,*k*,1) convolutional encoder

III. Analysis of Construcion Mechanism

With only a delay unit $\boldsymbol{D}$, a determinate information vector corresponds to a state in the (2*k*, *k*, 1) convolutional codes encoder. We suppose that $S_i=\boldsymbol{M}(t)$ denotes the current state, while $S_j=\boldsymbol{M}(t-1)$ is the previous state, where subscript *i* and *j* have the same meaning with (4)~(6). Clearly, altogether there are 2^k states in (2*k*, *k*, 1) convolutional codes because the size of the information vector are *k* bits in the $\boldsymbol{D}$, where any two states can mutually transfer, in other words, corresponding to trellis diagram, 2^k branches collect or branch in each state node. Fig. 2 shows the state transition of (6, 3, 1) convolutional codes, and a trellis diagram can be obtained when several such state transition connect with each other end-to-end.

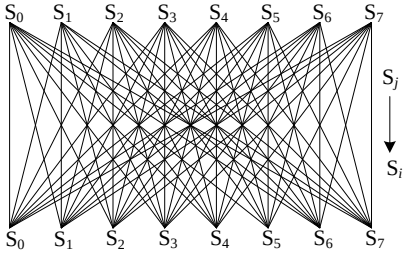


Figure 2. The state transition matrix of (6, 3, 1) convolutional codes

In order to explain the structure mechanism of (2*k*, *k*, 1) convolutional codes, we define a state transition matrix as follows:

$$\boldsymbol{C}=\boldsymbol{C}_1+\boldsymbol{C}_2=\begin{bmatrix}\boldsymbol{C}_1^{00} & \boldsymbol{C}_1^{01} & \ldots & \boldsymbol{C}_1^{0K}\\ \boldsymbol{C}_1^{10} & \boldsymbol{C}_1^{11} & \ldots & \boldsymbol{C}_1^{1K}\\ \vdots & \vdots & & \vdots\\ \boldsymbol{C}_1^{K0} & \boldsymbol{C}_1^{K1} & \ldots & \boldsymbol{C}_1^{KK}\end{bmatrix}+\begin{bmatrix}\boldsymbol{C}_2^{00} & \boldsymbol{C}_2^{01} & \ldots & \boldsymbol{C}_2^{0K}\\ \boldsymbol{C}_2^{10} & \boldsymbol{C}_2^{11} & \ldots & \boldsymbol{C}_2^{1K}\\ \vdots & \vdots & & \vdots\\ \boldsymbol{C}_2^{K0} & \boldsymbol{C}_2^{K1} & \ldots & \boldsymbol{C}_2^{KK}\end{bmatrix}$$
$$=\begin{bmatrix}\boldsymbol{C}^{00} & \boldsymbol{C}^{01} & \ldots & \boldsymbol{C}^{0K}\\ \boldsymbol{C}^{10} & \boldsymbol{C}^{11} & \ldots & \boldsymbol{C}^{1K}\\ \vdots & \vdots & & \vdots\\ \boldsymbol{C}^{K0} & \boldsymbol{C}^{K1} & \ldots & \boldsymbol{C}^{KK}\end{bmatrix} \quad (7)$$

where $K=2^k-1$, $\boldsymbol{C}_1^{ij}$, $\boldsymbol{C}_2^{ij}$, $\boldsymbol{C}^{ij}$ are elements of matrix $\boldsymbol{C}_1$, $\boldsymbol{C}_2$, $\boldsymbol{C}$, respectively. $\boldsymbol{C}^{ij}$ denotes the state transition from S_j to S_i, for example, when $\boldsymbol{M}(t-1)=011$ and $\boldsymbol{M}(t)=101$ of (6, 3, 1) convolutional codes, we can obtain the state transition that is $S_3\rightarrow S_5$, and the encoding output is $\boldsymbol{C}^{53}$. All state transitions of (2*k*, *k*, 1) convolutional codes are included in (7) which form a whole codeword space. The rows and columns in (7) are ordered by *i* and *j*, so the 2^k codes of each row (or column) correspond to not only the same $\boldsymbol{M}(t)$ (or $\boldsymbol{M}(t-1)$), but also the same state node where 2^k branches collect or branch in the trellis diagram. There are 2^{2k} elements in the matrix, which correspond to the 2^{2k} branches determined by the adjacent moments in the trellis diagram. Four properties about the code structure and distance property of (2*k*, *k*, 1) convolutional

codes are concluded with the help of computer programming for different k. Before proving these properties, we first list some lemmas.

Lemma 1: Define a 2^k-element natural binary codes set $\Phi(k)$ composed by all the k-bit binary vector $\boldsymbol{M}=[m_0\ m_1\ m_2\ \ldots\ m_{k-1}]^{\mathrm{T}}$. If all the elements in $\Phi(k)$ plus a certain same element respectively, the result will be mapped to themselves (without considering the sequence of elements).

Lemma 2: Take a bit in all elements of set $\Phi(k)$ with same position, the numbers of 0s and 1s are both 2^{k-1}.

Lemma 3: $\boldsymbol{M}\in\Phi(k)$, define $w(\boldsymbol{M})$ as the code weight of $\boldsymbol{M}$, then after $\boldsymbol{M}$ traverses $\Phi(k)$, the numbers that $w(\boldsymbol{M})$ is odd and even number are both 2^{k-1}.

Lemma 4: Delete k–s bits in the same position of all elements in set $\Phi(k)$, and form a new set $\Phi'(k)$(define a element of set $\Phi'(k)$ is $\boldsymbol{M}'_i=[m'_{i0}\ m'_{i1}\ m'_{i2}\ \ldots\ m'_{i(s-1)}]^{\mathrm{T}}$), where $0<s<k$, then that contains 2^{k-s} $\Phi(s)$ in $\Phi'(k)$.

Lemma 5: Take the k_1-bit vector $\boldsymbol{M}_1$ in $\Phi(k_1)$, and merge it with a k_2-bit vector $\boldsymbol{M}_2$ in $\Phi(k_2)$ as a (k_1+k_2)-bit vector. After $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ traverse $\Phi(k_1)$ and $\Phi(k_2)$ in order, we can obtain a new set $\Phi(k_1+k_2)$.

Lemma 1- Lemma 5 can be inferred from the natural binary code structure, so the proof will not be given here.

Lemma 6: Codeword $\boldsymbol{D}_1$, $\boldsymbol{D}_2$, $\boldsymbol{D}_3\in\Phi(k)$,then the code distance $d(\boldsymbol{D}_1+\boldsymbol{D}_3,\boldsymbol{D}_2+\boldsymbol{D}_3)=d(\boldsymbol{D}_1,\boldsymbol{D}_2)$.

Proof: It do not change the corresponding bit of $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ when one of the bits in $\boldsymbol{D}_3$ is 0. The code distance will not be changed though the corresponding bit is changed when one of the bits in $\boldsymbol{D}_3$ is 1, because the changes for $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ occur at the same time.

Lemma 7: The constant vector $\boldsymbol{M}=[m_0\ m_1\ m_2\ \ldots\ m_{k-1}]^{\mathrm{T}}\in\Phi(k)$, $\boldsymbol{P}=[p_{xy}]$ is a $k\times k$ constant matrix in GF(2), where x, y=0~k–1, the elements in any row (column) cannot be all 0 in $\boldsymbol{P}$. We define

$$\boldsymbol{H}_i=\boldsymbol{P}\times\boldsymbol{M}_i+\boldsymbol{M}=[h_{i0}\ h_{i1}\ h_{i2}\ \ldots\ h_{i(k-1)}]^{\mathrm{T}} \tag{8}$$

When $\boldsymbol{M}_i$ traverses $\Phi(k)$, all $\boldsymbol{H}_i$ can form a set $\psi(k)$. Take a bit in all elements of set $\psi(k)$ with same position, then the quantities of 0s and 1s are equal.

Proof: Suppose $\boldsymbol{H}'_i=\boldsymbol{P}\times\boldsymbol{M}_i=[h'_{i0}\ h'_{i1}\ h'_{i2}\ \ldots\ h'_{i(k-1)}]^{\mathrm{T}}$, then the first bit of $\boldsymbol{H}'_1$ is

$$\begin{aligned} h'_{i0} &= [p_{00}\ p_{01}\ \ldots\ p_{0(k-1)}]\times[m_{i0}\ m_{i1}\ m_{i2}\ \ldots\ m_{i(k-1)}]^{\mathrm{T}} \\ &= \sum_{y=0}^{k-1} p_{0y}m_{iy} \end{aligned} \tag{9}$$

When the coefficient p_{0y}=0, it equals that m_{iy} is deleted, from the definition of $\boldsymbol{M}'_i$ in Lemma 4, we can obtain

$$h'_{i0}=\sum_{y=0}^{k-1}p_{0y}m_{iy}=\sum_{y=0}^{s-1}m'_{iy}=w(\boldsymbol{M}'_i)(\mathrm{mod}\,2) \tag{10}$$

Combining Lemma 3 and Lemma 4, after i traverses 0~2^k -1, the number of 0 and 1 for h'_{i0} are equivalent. The first bit m_0 of $\boldsymbol{M}$ can be 0 or 1, from (8) we can obtain

$$h_{i0}=h'_{i0}+m_0=h'_{i0}\ or\ \overline{h'_{i0}} \tag{11}$$

When m_0=1, it equals to calculate the complement code of $h'_{00}\sim h'_{(2^k-1)0}$ simultaneously, that is 0 and 1 make interconversion with each other, and the interconversion number is in the same, so the number of 0 and 1 will keep equivalent. Similarly, $h_{i1}\sim h_{i(k-1)}$ can be proved.

Property 1: All the elements in the state transition matrix of (2k, k, 1) convolutional codes form the set $\Phi(2k)$.

Proof: Let the higher k bits of $\boldsymbol{C}^{ij}$ in (7) are

$$\boldsymbol{C}_h^{ij}=\boldsymbol{M}(t)+\boldsymbol{M}(t-1) \tag{12}$$

Considering the closure of GF (2), for any $\boldsymbol{M}(t\text{-}1)\in\Phi(k)$, we can always find the only $\boldsymbol{M}(t)\in\Phi(k)$ to make sure $\boldsymbol{C}_h^{ij}\in\Phi(k)$ is a given vector. When $\boldsymbol{M}(t\text{-}1)$ traverses $\Phi(k)$, after the vector encoded, it has a unchanged parity bit $\boldsymbol{P}\times(\boldsymbol{M}(t)+\boldsymbol{M}(t\text{-}1))$, the lower k bits can be obtained by Lemma 1

$$\boldsymbol{C}_l^{ij}=\boldsymbol{P}\times(\boldsymbol{M}(t)+\boldsymbol{M}(t-1))+\boldsymbol{M}(t-1) \tag{13}$$

where (13) can compose a set $\Phi(k)$. Known from Lemma 5, after $\boldsymbol{C}_h^{ij}$ traverses $\Phi(k)$, we can get $\Phi(2k)$.

Property 2: The distance distribution of 2^k elements in any row of the state transition matrix for (2k, k, 1) convolutional codes is as same as that of embedded codes.

Proof: $\boldsymbol{C}_1^{i0}\sim\boldsymbol{C}_1^{iK}$, $\boldsymbol{C}_2^{i0}\sim\boldsymbol{C}_2^{iK}$, $\boldsymbol{C}^{i0}\sim\boldsymbol{C}^{iK}$ are the 2^k elements in a certain row of $\boldsymbol{C}_1$, $\boldsymbol{C}_2$, $\boldsymbol{C}$, respectively. For $\boldsymbol{C}_1^{i0}\sim\boldsymbol{C}_1^{iK}$, obviously, the 2^k elements have a same $\boldsymbol{M}(t)$, and $\boldsymbol{M}(t\text{-}1)$ traverses $\Phi(k)$. From Lemma 1 we can know, $\boldsymbol{M}(t)+\boldsymbol{M}(t\text{-}1)$ composes $\Phi(k)$, $\boldsymbol{C}_1^{i0}\sim\boldsymbol{C}_1^{iK}$ is the code set of embedded codes according to (4). And for $\boldsymbol{C}_2^{i0}\sim\boldsymbol{C}_2^{iK}$, the 2^k elements have a same $\boldsymbol{M}(t)$, so $\boldsymbol{C}_2^{i0}=\boldsymbol{C}_2^{i1}=\ldots=\boldsymbol{C}_2^{iK}$. It can be known that $\boldsymbol{C}^{i0}\sim\boldsymbol{C}^{iK}$ is a new codes set composed by the sum of embedded codes and a same element. In Lemma 6, the code distance between two codes will not change if they plus a same code at the same time. It can be known that the new codes set distance distribution is as same as that of embedded codes.

Property 3: The numbers of 0s and 1s in the same bit of the 2^k codes in any row (column) of (2k, k, 1) convolutional codes state transition matrix are equal.

Proof: The higher k bits $\boldsymbol{C}_h^{i0}\sim\boldsymbol{C}_h^{iK}$ (or $\boldsymbol{C}_h^{0j}\sim\boldsymbol{C}_h^{Kj}$) in any row(column) of the state transition matrix can compose $\Phi(k)$. The conclusion can be proved by Lemma 2. Next we consider the lower k bits $\boldsymbol{C}_l^{0j}\sim\boldsymbol{C}_l^{Kj}$ in a certain column. Known from (12), P and $\boldsymbol{M}(t\text{–}1)$ are the same. $\boldsymbol{M}(t)+\boldsymbol{M}(t\text{-}1)$ composes $\Phi(k)$, which satisfies (8). The conclusion can also be proved by Lemma 7. The lower k bits $\boldsymbol{C}_l^{i0}\sim\boldsymbol{C}_l^{iK}$ is considered at last, the (13) can be rewritten as

$$\begin{aligned} \boldsymbol{C}_l^{ij} &= \mathrm{P} \times (\boldsymbol{M}(t) + \boldsymbol{M}(t-1)) + \boldsymbol{M}(t-1) \\ &= (\mathrm{P} + \mathrm{I})\boldsymbol{M}(t-1) + \mathrm{P} \times \boldsymbol{M}(t) \end{aligned} \quad (14)$$

$\boldsymbol{M}(t\text{-}1)$ composes $\Phi(k)$, so the formula still satisfies (8), and the conclusion is proved.

Property 4: The code weight sum of the 2^k codes in any row (column) of the state transition matrix for ($2k$, k, 1) convolutional codes is equal to $2^k \times k$.

Proof: According to Property 3, we can obtain a weak conclusion that the number of 0 and 1 in the 2^k codes in any row (column) are equivalent in (7). So the code weight sum of the 2^k codes is

$$w = \sum_{i=0}^{2^k-1} w\left(\boldsymbol{C}^{ij}\right) = \sum_{j=0}^{2^k-1} w\left(\boldsymbol{C}^{ij}\right) = \frac{2k \times 2^k}{2} = k \times 2^k \quad (15)$$

Property 5: In decimal case, the sum of elements in any certain row (column) in the state transition matrix of ($2k$, k, 1) convolutional codes is constant.

Proof: The calculation is limited to GF(∞). Suppose that $\boldsymbol{C}^{ij} = [c_{2k-1}^{ij}\ c_{2k-2}^{ij} \ldots c_1^{ij}\ c_0^{ij}]^{\mathrm{T}}$, the decimal sum of the 2^k codes in any certain column in (7) is

$$S = \sum_{i=0}^{2^k-1} \boldsymbol{C}^{ij} = \sum_{i=0}^{2^k-1} \sum_{n=0}^{2k-1} 2^n c_n^{ij} = \sum_{n=0}^{2k-1} \sum_{i=0}^{2^k-1} 2^n c_n^{ij} \quad (16)$$

Known from Property 3, when i traverses 0~2^k-1, the numbers of 0s and 1s in c_n^{ij} are both 2^{k-1}, so

$$S = \sum_{n=0}^{2k-1} \sum_{i=0}^{2^k-1} 2^n c_n^{ij} = \sum_{n=0}^{2k-1} 2^n 2^{k-1} = 2^{k-1}(2^{2k} - 1) \quad (17)$$

Similarly, it can be proved that the decimal sum of the 2^k codes in any certain row is also $2^{k-1}(2^{2k}-1)$.

Among the four properties above, Property 1 shows that there are no same elements in the state transition matrix, in other words, no same codeword in the 2^{2k} branches of the trellis diagram, this is important for improving the distance characteristics. Property 2 shows that for any single state node, the distance distribution of the 2^k branches collecting in this node is as same as that of embedded codes. So the higher quality the embedded codes have, the easier it is to structure a good ($2k$, k, 1) code. A class of ($2k$, k) double loop cyclic linear block codes with a minimum and maximum distance is introduced as the embedded codes in [14]. The minimum distance when k=2~8 and $\boldsymbol{P}$ matrix in (3) are given in Table. I. The k-bit binary column vectors of $\boldsymbol{P}$ matrix are converted to octonary number for convenience. We take the (6, 3) double loop cyclic linear block codes as an example, according to Table 1, its generator matrix is

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} \\ [3\ 5\ 6]_{\mathrm{O}} \end{bmatrix} = \begin{bmatrix} 100011 \\ 010101 \\ 001110 \end{bmatrix}_{\mathrm{B}}^{\mathrm{T}} \quad (18)$$

We find that $\boldsymbol{I}$ or $\boldsymbol{P}$ is circular in (18), but $\boldsymbol{G}$ not, this is just the origin of the so-called *double loop cyclic*. We can easily obtain code set of (6, 3) codes base on $\boldsymbol{G}$, which are 000000, 001110, 010101, 011011, 100011, 101101, 110110 and 111000. Clearly, its minimum distance that is 3 shows a high performance-price ratio under the length of codes $2k$=6.

TABLE I. MATRIX P AND MINIMUM DISTANCE OF ($2K$, K) EMBEDDED CODES

k	$\boldsymbol{P}$	Minimum distance
2	[1 2]	2
3	[3 5 6]	3
4	[13 15 16 7]	4
5	[7 23 31 34 16]	4
6	[16 7 43 61 70 34]	4
7	[154 66 33 146 63 131]	4
8	[164 72 35 216 107 243 321 350]	5

In Property 3-5, it is shown in the trellis diagram that the code structure and weight distribution of ($2k$, k, 1) convolutional codes are well balanced, which is very helpful for obtaining the high quality code. Besides, Property 5 shows an interesting magic square characteristics[15], and considering the coding constraint degree is 1, the ($2k$, k, 1) convolutional codes are named by "*first-order magic square convolutional codes*". In order to analyze the structure mechanism of ($2k$, k, 1) convolutional codes with the help of Property 1~5 further, we take the (6, 3, 1) convolutional codes as an example. Firstly, suppose that there are no embedded zeroes and superposition, $\boldsymbol{C}_1$ is outputted as the final codes, according to (4) and (18), we can obtain the first term of (7) as in (19).

$$\boldsymbol{C}_1 = \begin{bmatrix} 000000\ 001110\ 010101\ 011011\ 100011\ 101101\ 110110\ 111000 \\ 001110\ 000000\ 011011\ 010101\ 101101\ 100011\ 111000\ 110110 \\ 010101\ 011011\ 000000\ 001110\ 110110\ 111000\ 100011\ 101101 \\ 011011\ 010101\ 001110\ 000000\ 111000\ 110110\ 101101\ 100011 \\ 100011\ 101101\ 110110\ 111000\ 000000\ 001110\ 010101\ 011011 \\ 101101\ 100011\ 111000\ 110110\ 001110\ 000000\ 011011\ 010101 \\ 110110\ 111000\ 100011\ 101101\ 010101\ 011011\ 000000\ 001110 \\ 111000\ 110110\ 101101\ 100011\ 011011\ 010101\ 001110\ 000000 \end{bmatrix} \quad (19)$$

It is easy to verify that (19) follows Property 2-5, but Property 1. We find that 8 code-words of each row (column) are exactly the same with the codeset of (6, 3) embedded codes, the distance characteristics of individual row (column) seems to be good. However, the code space is only $\Phi(3)$ in (19), each codeword appears in the matrix repeatedly for 2^3=8 times, which means that there are 8 branches with the same codeword will appear frequently in Fig. 2, the distance characteristics will be badly affected because the code distance will be 0 frequently when calculating the free distance, and we cannot obtain excellent codes by $\boldsymbol{C}_1$ though it has finished the encoding with memory. Fortunately, these problems could be solved if we consider the embedded zero, according to (5), we can obtain the second term of (7) as in (20).

$$\boldsymbol{C}_2 = \begin{bmatrix} 000000\ 000000\ 000000\ 000000\ 000000\ 000000\ 000000\ 000000 \\ 000001\ 000001\ 000001\ 000001\ 000001\ 000001\ 000001\ 000001 \\ 000010\ 000010\ 000010\ 000010\ 000010\ 000010\ 000010\ 000010 \\ 000011\ 000011\ 000011\ 000011\ 000011\ 000011\ 000011\ 000011 \\ 000100\ 000100\ 000100\ 000100\ 000100\ 000100\ 000100\ 000100 \\ 000101\ 000101\ 000101\ 000101\ 000101\ 000101\ 000101\ 000101 \\ 000110\ 000110\ 000110\ 000110\ 000110\ 000110\ 000110\ 000110 \\ 000111\ 000111\ 000111\ 000111\ 000111\ 000111\ 000111\ 000111 \end{bmatrix} \tag{20}$$

Adding it and (19) together the result is

$$\boldsymbol{C} = \boldsymbol{C}_1 + \boldsymbol{C}_2 = \begin{bmatrix} 000000\ 001110\ 010101\ 011011\ 100011\ 101101\ 110110\ 111000 \\ 001111\ 000001\ 011010\ 010100\ 101100\ 100010\ 111001\ 110111 \\ 010111\ 011001\ 000010\ 001100\ 110100\ 111010\ 100001\ 101111 \\ 011000\ 010110\ 001101\ 000011\ 111011\ 110101\ 101110\ 100000 \\ 100111\ 101001\ 110010\ 111100\ 000100\ 001010\ 010001\ 011111 \\ 101000\ 100110\ 111101\ 110011\ 001011\ 000101\ 011110\ 010000 \\ 110000\ 111110\ 100101\ 101011\ 010011\ 011101\ 000110\ 001000 \\ 111111\ 110001\ 101010\ 100100\ 011100\ 010010\ 001001\ 000111 \end{bmatrix} \tag{21}$$

where the code space is expanded to $\Phi(6)$, only first row (column) is the same with the codeset of (6, 3) embedded codes. Although there is a difference between first row (column) and other rows (columns) in (21), they can hold equivalent code distance, see Property 2. For example, the third row of (21) is

$$\boldsymbol{C}^{2j} = \begin{bmatrix} 010111 \\ 011001 \\ 000010 \\ 001100 \\ 110100 \\ 111010 \\ 100001 \\ 101111 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 010101 \\ 011011 \\ 000000 \\ 001110 \\ 110110 \\ 111000 \\ 100011 \\ 101101 \end{bmatrix}^{\mathrm{T}} + \begin{bmatrix} 000010 \\ 000010 \\ 000010 \\ 000010 \\ 000010 \\ 000010 \\ 000010 \\ 000010 \end{bmatrix}^{\mathrm{T}} \tag{22}$$

where 000010 are added to the embedded codes, but the code distances don't change. By this way, the same codes are avoided in (21) under ensuring the distance characteristics of all rows (columns), and the free distance of codes can be significantly improved. From the analysis above, we can find that embedded zero and superposition in Fig.1 play a very important role to construct the (2*k*, *k*, 1) convolutional codes, because they can keep Property 2-5 while obtain Property 1.

In addition, converting to decimal number for all codewords of (21), we can obtain a magic square matrix as follows

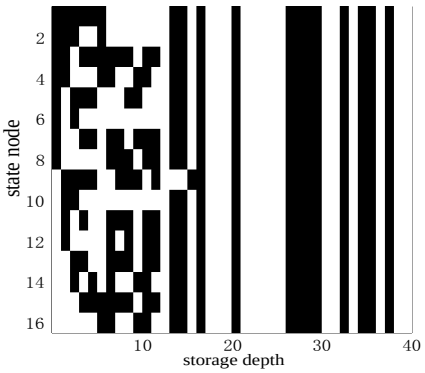
$$\boldsymbol{C} = \begin{bmatrix} 0\ 14\ 21\ 27\ 35\ 45\ 54\ 56 \\ 15\ 1\ 26\ 20\ 44\ 34\ 57\ 55 \\ 23\ 25\ 2\ 12\ 52\ 58\ 33\ 47 \\ 24\ 22\ 13\ 3\ 59\ 53\ 46\ 32 \\ 39\ 41\ 50\ 60\ 4\ 10\ 17\ 31 \\ 40\ 38\ 61\ 51\ 11\ 5\ 30\ 16 \\ 48\ 62\ 37\ 43\ 19\ 29\ 6\ 8 \\ 63\ 49\ 42\ 36\ 28\ 18\ 9\ 7 \end{bmatrix}_{\mathrm{D}} \tag{23}$$

where we can verify one by one, that the accumulation of each (columns) is $2^2 \times (2^6-1) = 252$, the result is in accordance with (17).

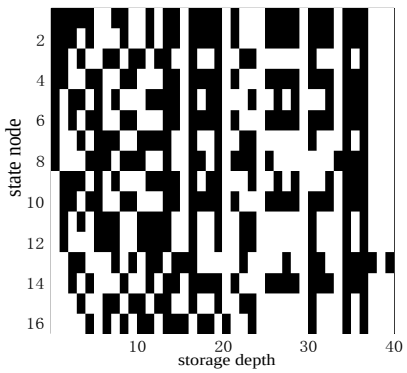
IV. Simulation

A matrixing method is proposed to perform Viterbi algorithm for traditional (2, 1, k) convolutional codes in [13], the advantage of matrixing is that a single-structured decoder with parallel processing ability [16] can be designed, this decoding scheme can also be used in (2k, k, 1) convolutional codes in the Gaussian channel and BPSK modulation. In simulation, double precision data is used to calculate path metric, the storage depth of survived path is τ=10k, and simulation should be stopped when 2000~10000 mistakes appear.

In order to monitor the decoding process, we can observe the output of survived path memory by matrix viewer. Fig. 3 shows the screen of (8, 4, 1) convolutional codes. where the storage depth τ is set to 10×k=40, the numbers of state node is 2^k=16, and the black and white denote 1 and 0, By comparing the two figures, we see the decoder can get the survivor path together better in (a), it can implement a helpful observation for real time channel condition.



(a)



(b)

Figure3. Path storage matrix of (8, 4, 1) convolutional codes: (a) Good channel condition, (b) Bad channel condition

Take Table. I as embedded codes, we can construct seven (2k, k, 1) convolutional codes. Fig. 4 shows their BER performance. When k=2~8, we can see that 3~6dB coding gain can be obtained compared with no coding at BER=10^{-5}. In addition, with k increases, the gain increment of BER performance gradually reduces, and it has a similar convergence with the conventional convolutional codes.

It is well known that the state numbers of (n, k, l) convolutional codes are 2^{kl}, therefore, there are 2^k states in both (2k, k, 1) and (2, 1, k) convolutional codes. As a result, these two codes need perform 2^k times add-compare-select (ACS) as well as save-update for survivor path in the Viterbi matrix decoding, the decoding complexity is approximately equivalent, and the error-correcting capability can be made a comparison. To enhance convincing, five optimal (2, 1, k) codes are chosen from [14], the BER performance of the two types codes when k=2~6 is shown in Fig. 5. When BER=10^{-5}, we can see that (2k, k, 1) convolutional codes have a 0.2~0.5 dB advantage over (2, 1, k) codes on coding gain except when k=2, 3.

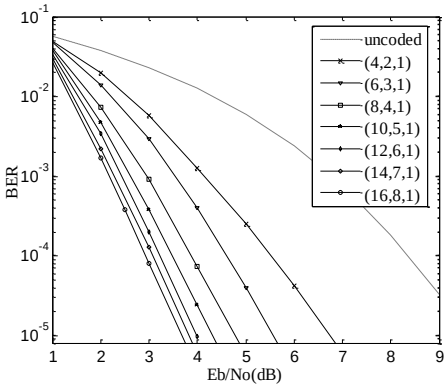


Figure 4. Error performance of (2k, k, 1) convolutional codes

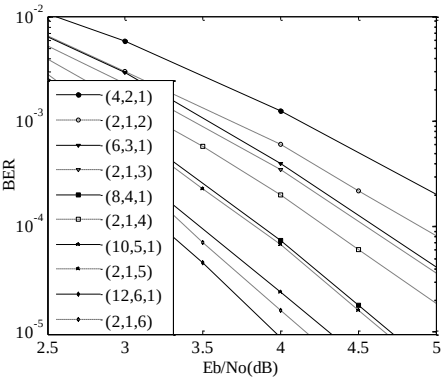


Figure 5. Error performance comparison between (2k, k, 1) and (2, 1, l) convolutional codes

TABLE II. Comparison of Two Convolutional Codes Decoding Speed

k	(2k, k, 1)	(2, 1, k)
2	1.8×10^7	4.8×10^6
3	1.7×10^7	4.9×10^6
4	1.6×10^7	4.2×10^6
5	1.1×10^7	3.8×10^6
6	6.4×10^6	2.9×10^6
7	5.6×10^6	2.6×10^6
8	2.2×10^6	1.3×10^6

In addition, we will also discuss the efficiency of (2*k*, *k*, 1) and (2, 1, *k*) convolutional codes. The branches that join the same state node for (2*k*, *k*, 1) and (2, 1, *k*) convolutional codes are 2^k and 2, see Fig. 2, so the implementation of ACS should be 1-from-2^k and 1-from-2, respectively, it led to (2*k*, *k*, 1) convolutional codes seem to be more complex. However, we find that the information bits processed for (2*k*, *k*, 1) and (2, 1, *k*) convolutional codes are respectively *k* and 1 in each decoding time, the batch processing for the former is more efficient. In other words, to obtain *k*-bit information, *k*-times decoding must be implemented for (2, 1, *k*) convolutional codes, this process will still result in 2^k times 1-from-2, from this we can see that (2*k*, *k*, 1) convolutional codes is essentially more efficient. Table 2 shows the bits of information calculated after the program run for 10 minutes for the two types of codes. It is obvious that the (2*k*, *k*, 1) convolutional codes have a higher decoding speed.

V. Conclusion

A new class of (2*k*, *k*, 1) convolutional codes is constructed by combining with the (2, 1, 1) convolutional codes and the (2*k*, *k*) double loop cyclic codes, its main contribution is present a novel method to increase the memory length of convolutional codes. The distance property and the structural mechanism of the codes are revealed by defining a state transition matrix as well as trellis diagram. The five properties of the state transition matrix are proved by algebraic method, which indicate that the codes structure is very excellent in both proportionality and diversity. Simulation results show the BER performance of parts of codes, and the advantages of the codes in error-correcting capability and decoding speed is verified further. Similar to the concatenated codes, (2*k*, *k*, 1) convolutional codes also use the existing codes, however, the structure mechanism is quite different with that of concatenated codes. The significance of the research lies in a new method that constructs long codes with short codes is proposed, and a broad development space is to be expanded in this field. We will conduct further research on how to construct (2*k*, *k*, *l*) convolutional codes based on this paper in order to increase both block length and constraint degree.

References

[1] J. P. Robinson, "A class of binary recurrent codes with limited error propagation," *IEEE Transactions on Informational Theory*, vol. 13, pp. 106-113, Jan 1967.

[2] J. L. Massey and D. J. Costello, "Nonsystematic convolutional codes for sequential decoding in space applications," *IEEE Transactions on Communication Technology*, vol. 19, pp. 806-813, Oct 1971.

[3] B. Chen and C. Sundberg, "Complementary punctured-pair convolutional codes for digital audio broadcasting," *IEEE Transactions on Communicatio*n, vol. 48, pp. 1829-1839, Nov 2000.

[4] S. Lin, J. Daniel, and Costello, *Error Control Coding: Fundamentals and Applications*, 2nd ed. Pearson Education, 2004, pp. 515-558.

[5] H. Moon and D. C. Cox, "Performance of unequally punctured convolutional codes," *IEEE Transactions on Wireless Communications*, vol. 8, pp. 3903-3909, Aug 2009.

[6] S. Nagaraj and M. R. Bell, "Concatenated trellis coded modulation for quasi-static fading channels," *IEEE Transactions on Wireless Communications*, vol. 6, pp. 3220-3228, Sep 2007.

[7] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communication*, vol. 44, pp. 1261-1271, Oct 2007.

[8] R. M. Tanner, D. Sridhara, *et al*., "LDPC block and convolutional codes based on circulant matrices," *IEEE Transactions on Informational Theory*, vol. 50, pp. 2966-2984, Dec 2004.

[9] A. Katsiotis, P. Rizomiliotis, *et al.*, "New constructions of high-performance low-complexity convolutional codes," *IEEE Transactions on Communication*, vol. 58, pp. 1950-1961, Jul 2010.

[10] A. E. Pusane, R. Smarandache, *et al.*, "Deriving good ldpc convolutional codes from ldpc block codes," *IEEE Transactions on Informational Theory*, vol. 57, pp. 835-857, Feb 2011.

[11] B. S. Tan, K. H. Li, and K. C. The, "Performance analysis of LDPC codes with maximum-ratio combining cascaded with selection combining over nakagami-m fading," *IEEE Transactions on Wireless Communications*, vol. 10, pp. 1886-1894, Jun 2011.

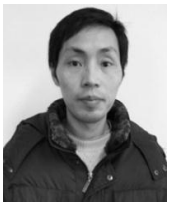
[12] G. Kowarzyk and N. Belanger, "Efficient search algorithm for determining optimal R=1/2 systematic convolutional self-doubly orthogonal codes," *IEEE Transactions on Communications*, vol. 60, pp. 3-8, Jan 2012.

[13] W. Q. Peng, X. B. Wu, and C. C. Zhang, "A matrix implementation scheme of viterbi decoder," *Journal of Circuits and Systems*, vol. 17, pp. 115-120, May 2012.

[14] X. M. Wang and G. Z. Xiao, *Error Correcting Codes-Principle and Method*, Xi'an: Xidian University Press, 2001, pp. 159-161.

[15] Z. L. Michael, L. Elizabeth, *et al*., "On nonsingular regular magic squares of odd order," *Linear Algebra and Its Applications*, vol. 437, pp. 1346-1355, Jan 2012.

[16] F. Sun and T. Zhang, "Low-power state-parallel relaxed adaptive viterbi decoder," *IEEE Transactions on Circuits and Systems*, vol. 54, pp. 1060-1068, May 2007.



Wanquan Peng was born in Chongqing, China in 1974. He received his M.S. degrees and Ph.D. degrees in Circuits and Systems from Chongqing University, Chongqing, China in 2005 and 2009, respectively. From 2005-2013, he is working in Chongqing Vocational Institute of Engineering. In the period 2004-2009 his interests was in the algorithm research and hardware design for product code. Currently he is working with new convolutional codes and LDPC convolutional codes. His research interests include error control coding and information theory.

Chengchang Zhang is with the Laboratory of Aerocraft Tracking Telemetering & Command and Communication, Ministry of Education, Chongqing University. He received the M.S. degree in Communication and information systems in 2005 and the Ph.D. degree in 2011, from the Chongqing University. In 1998, he joined the Chinese Institute of Electronics, CIE. His main research interests include error correction coding, software radio and Multi-FPGA systems design.

