

Chapter 6

Trellis-pruned and Punctured convolutional codes

6.1 Trellis-pruned convolutional codes

Trellis-pruned convolutional codes (determinate state convolutional codes) [8] have been used to achieve unequal error protection (UEP) in many practical applications. But it was not used in the iterative decoding systems. How it performs in the BICM-ID system is very interesting. Through the trellis pruning, the minimum free distance of the code will be enhanced. In this chapter, we design the trellis-pruned convolutional codes with EXIT chart. Besides, we found that if the bit mapping changes along with the trellis-pruning convolutional codes, the performance of them will be enhanced very much compared to the original convolutional codes.

6.1.1 Encoding of the trellis-pruned convolutional codes

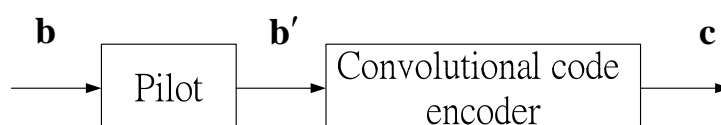


Fig. 6-1: Trellis-pruned convolutional encoder

To construct a trellis-pruned convolutional codes, we add some determinate bits (in the following we will call them pilot bits) into source data. If we add one bit for every K information bits and pass the pilot-inserted bits to a convolutional encoder, we will have the coded bits as follows.

$$b_0, b_1, \dots, b_{K-1}, p \quad \Rightarrow \quad \begin{matrix} c_0^0 & c_1^0 & \cdots & c_{K-1}^0 & v^0 \\ c_0^1 & c_1^1 & \cdots & c_{K-1}^1 & v^1 \end{matrix}$$

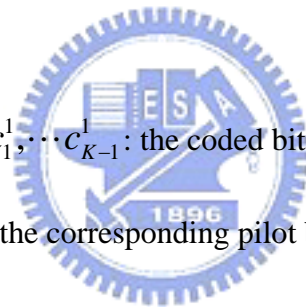
where

b_0, b_1, \dots, b_{K-1} : the K information bits

p : the inserted pilot bit

$c_0^0, c_1^0, \dots, c_{K-1}^0$ and $c_0^1, c_1^1, \dots, c_{K-1}^1$: the coded bits of b_0, b_1, \dots, b_{K-1}

v_0, v_1 : the coded bits of the corresponding pilot bit



The actual code rate after adding pilot bits is $R \cdot K / (K+1)$ and we have $R / (K+1)$ code rate loss. Generally, we can add arbitrary number of bits for every K information bits. This can be simulated by EXIT chart.

6.1.2 Decoding of the trellis-pruned convolutional codes

The inserted pilot bits can be zeros or ones, and we use all zero pilot bits in this

chapter. Fig.6-2 shows the decoding trellises of the original code and the trellis-pruned code respectively. Here we use a constraint length 3 convolutional code as example. Obviously, the trellis after trellis pruning is changed.

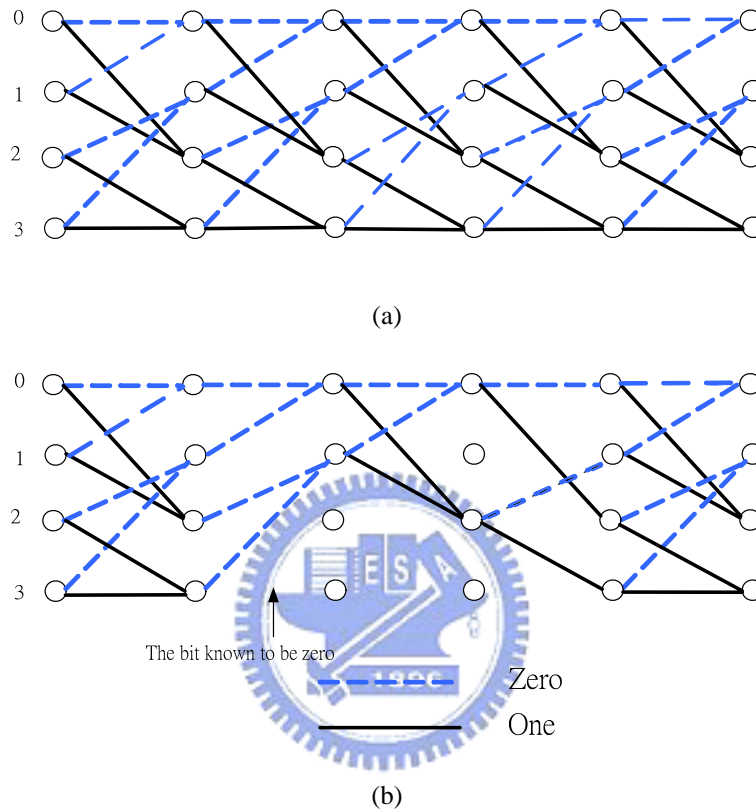


Fig. 6-2: Decoding trellis (a) the original convolutional codes (b) the trellis-pruned convolutional code

When decoding for a zero determinate pilot bit, the decoding paths will be reduced to four paths as Fig.6-2 (b) shows. This bit will affect the following two stages of decoding processing. By pruning away the “wrong” codeword path, the pruned-code may perform better compared to the original code.

6.1.3 Decoder transfer characteristics with trellis pruning

The decoder transfer characteristics with path-pruned is depicted in Fig.6-3.

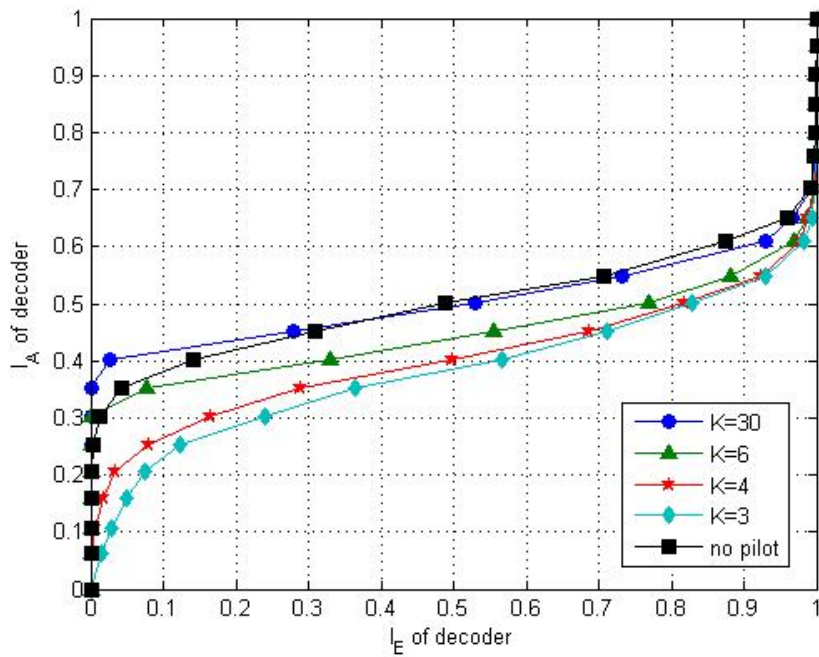


Fig. 6-3: Transfer characteristics of the decoder with $K=3,4,6,30$ and the original code, the generator

$$\text{polynomial is } (g_0, g_1) = (133_8, 171_8).$$

In Fig.6-3, we can see that when K is decreased from 30 to 3, the decoder transfer characteristics will be lower and sharper. Especially, when K decreases from 6 to 4, the curves at low I_E , about 0 to 0.2, have big changes. Another observation is that it is almost the same value of input I_A when I_E is near one for all curves, see Fig. 6-4. So I think that if we choose $K=4$, we will have a lot of improvement in the performance.

In the BICM-ID systems, the soft information exchanges between the detector and the decoder. Different detectors will result in different performances. Here, we fix the detector as APP detector but change the bit mappings, e.g. Natural, Gray, Anti-Gray and Type A, B, C.

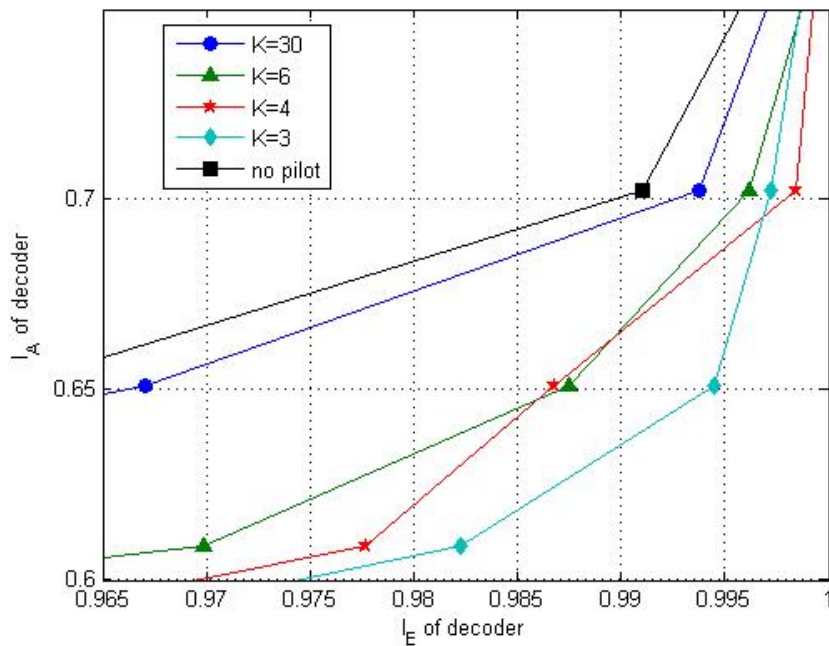


Fig. 6-4: The zoom in vision of Fig.6-3.

In Fig. 6-5, we can see that the detector with Type B mapping in 16QAM has the smallest SNR about 2 dB and others are larger than 2 dB. Therefore, the most suitable mapping for the trellis-pruned $(133_8, 171_8)$ convolutional codes with $K=4$ is Type B mapping in 16QAM.

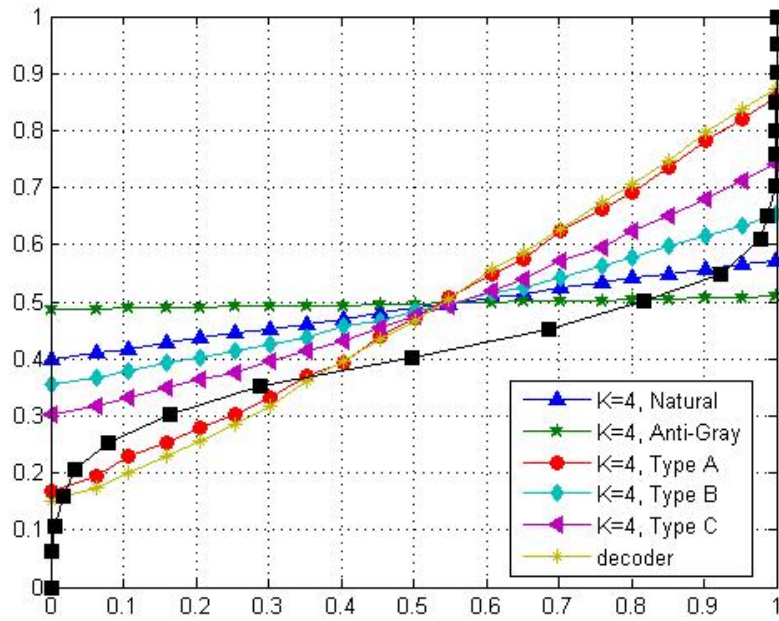


Fig. 6-5: EXIT chart for different bit-mappings in 16QAM BICM-ID systems at $E_b / N_0 = 2dB$.

6.1.4 Performance



In Fig.6-6, we show the performance of BICM-ID with K=4 trellis-pruned $(133_8, 171_8)$ convolutional code under 20 iterations. As our prediction, the most suitable labeling between the above six labelings is Type B labeling. If we have not labeling designs, the most suitable labeling for the $(133_8, 171_8)$ convolutional code in BICM-ID systems is Anti-Gray (3.7 dB) and the improvement on performance after labeling designing is about $3.7-1.9=1.8$ dB.

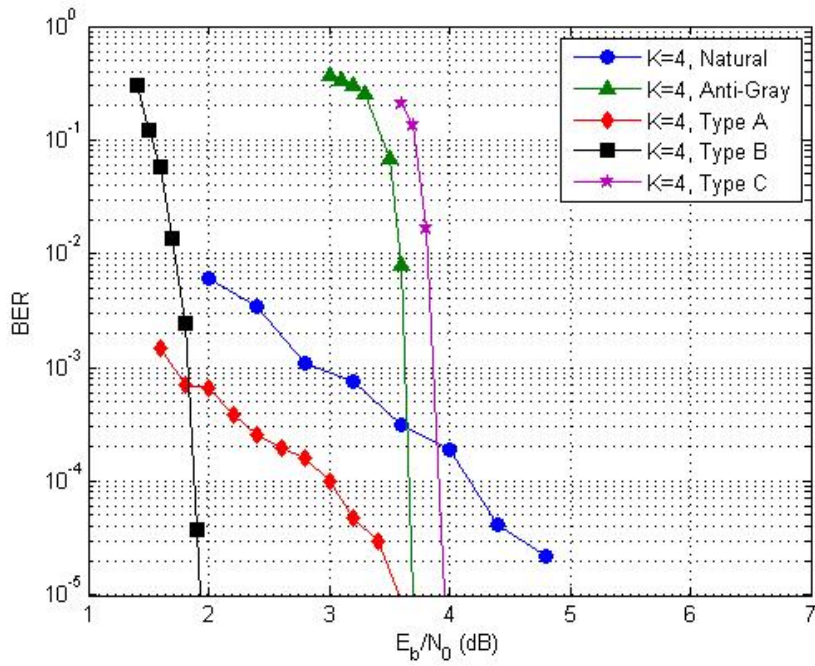


Fig. 6-6: BER of BICM-ID with six mappings with the $K=4$ trellis-pruned $(133_8, 171_8)$ convolutional

code, 16QAM, 20 iterations.

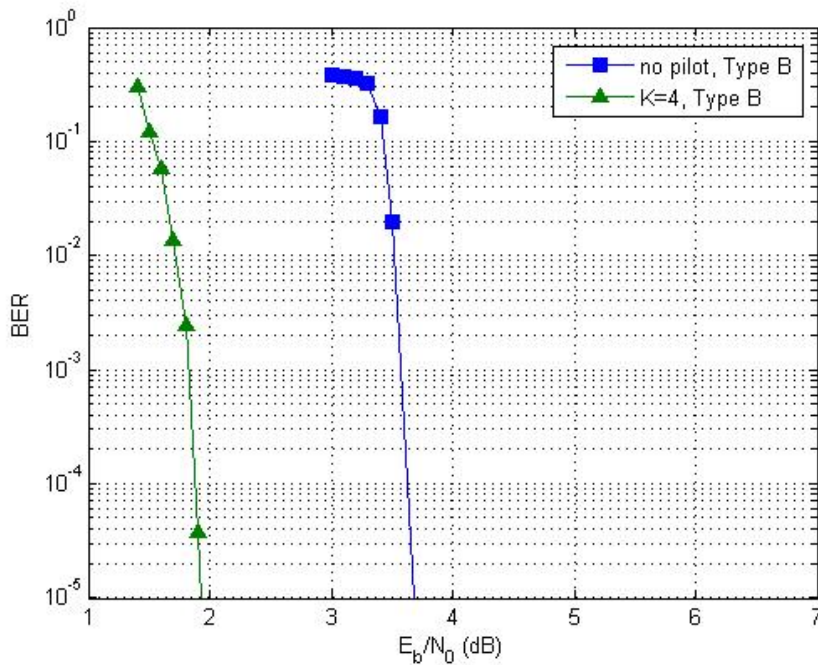


Fig. 6-7: BER of the BICM-ID systems with trellis-pruned and original convolutional codes, 16QAM,

20 iterations, both are Type B labeling.

Fig. 6-7 shows the performances of the BICM-ID systems with trellis-pruned and original $(133_8, 171_8)$ convolutional codes. The labelings used are both Type B labeling. The performance improvement after trellis pruning is about 1.7 dB.

6.2 Punctured convolutional codes

Convolutional codes with Viterbi decoding are the most attractive means to achieve significant coding gains on discrete noise channels. The $R=1/2$ convolutional code with generator polynomials $(g_0, g_1) = (133_8, 171_8)$ and constraint length 7 is the industry-standard code. In order to achieve higher transmission rate and have good error performance, one way is to use powerful high-rate $R=k/n$ convolutional codes. For high-rate $R=k/n$ codes, the number of operations and the amount of memory path histories to be stored increase rapidly and the implementation of the Viterbi decoder becomes impractical. Punctured convolutional codes [20] are used to simplify the Viterbi decoding. New good punctured convolutional codes are discovered [21], [10]. In this section, first we will describe the methods to choose the puncturing patterns in [10], [20], [21], then we will utilize EXIT chart to design the puncturing patterns.

6.2.1 Minimum free distance

High-rate $R = k/n$ convolutional codes can be obtained by a mother code with code rate $R = 1/n$. Here we use the mother codes with $R = 1/2$. By “steal” the coded bits at the output of the encoder, variable rate convolutional codes are produced.

It has been shown that the upper bound on the error event and bit error probabilities of any rate k/n codes with Viterbi decoding on discrete memoryless channels [22]-[24]

are given by

$$P \leq \frac{1}{k} \sum_{d=d_f}^{\infty} a_d P_d \quad (6.1)$$

and

$$P_b \leq \frac{1}{k} \sum_{d=d_f}^{\infty} c_d P_d \quad (6.2)$$

where

d_f : the minimum free distance of the code

a_d : the number of incorrect paths of Hamming weight d for $d \geq d_f$ that

diverge from the correct path and re-emerge with it at some later stage

c_d : the total number of error bits produced by the incorrect paths (error coefficient)

P_d : the probability of picking an incorrect path in the Viterbi decoding process

The criterion of a good punctured code is to choose a puncturing pattern with maximum free distance d_f and minimum total number of bit errors c_d .

A punctured code is generated from a rate-1/2 mother code using puncturing matrix \mathbf{P} of period p as follows,

$$\mathbf{P} = \begin{pmatrix} g_{11} & \cdots & g_{1p} \\ g_{21} & \cdots & g_{2p} \end{pmatrix} \quad (6.3)$$

In the following, the code rate-2/3 and 3/4 puncturing designs will be described.

6.2.1.1 Puncturing design on rate 2/3 codes

In this section, we show the puncturing pattern design of $(5_8, 7_8)$ and $(133_8, 171_8)$ convolutional codes both being punctured to rate-2/3. The puncturing matrix we use is of period 2.

$$\mathbf{P} = \begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix} \quad (6.4)$$

It is equivalent when we just shift the columns of the puncturing matrix.

Therefore we have two different puncturing matrices.

$$\mathbf{P}_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad \mathbf{P}_2 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad (6.5)$$

where 0 implies that the corresponding bit is punctured. Table 6-1 and 6-2 show the a_d and c_d of the two codes $(5_8, 7_8)$ and $(133_8, 171_8)$.

Code polynomial		R	puncturing matrix	$a_d, d=1,2,\dots$
5	7	2/3	\mathbf{P}_1	0, 0, 1, 4, 14, 40, ...
			\mathbf{P}_2	0, 0, 1, 10, 18, 38, ...
133	171	2/3	\mathbf{P}_1	0, 0, 0, 0, 1, 2, 17, 32, ...
			\mathbf{P}_2	0, 0, 0, 0, 0, 1, 15, 33, ...

TABLE 6-1: The number of the incorrect paths for the rate-2/3 punctured

$(5_8, 7_8)$ and $(133_8, 171_8)$ codes

Code polynomial		R	puncturing matrix	$c_d, d=1,2,\dots$
5	7	2/3	\mathbf{P}_1	0, 0, 2, 18, 86, 308, ...
			\mathbf{P}_2	0, 0, 2, 58, 132, 334, ...
133	171	2/3	\mathbf{P}_1	0, 0, 0, 0, 6, 7, 69, 154, ...
			\mathbf{P}_2	0, 0, 0, 0, 0, 5, 73, 163, ...

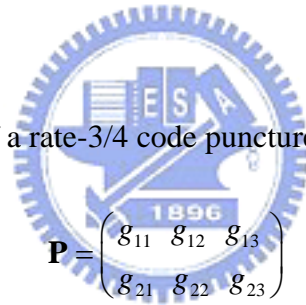
TABLE 6-2: The total number of the error bits produced by the incorrect paths for the rate-2/3

punctured $(5_8, 7_8)$ and $(133_8, 171_8)$ codes

The minimum free distance of rate-2/3 punctured $(5_8, 7_8)$ convolutional code is

2. For \mathbf{P}_1 and \mathbf{P}_2 , c_3 are both 2 but c_4 are 18 and 58 respectively. Therefore, the puncturing matrix we choose for $(5_8, 7_8)$ code is \mathbf{P}_1 . Choosing puncturing matrix for $(133_8, 171_8)$ convolutional code is easier than $(5_8, 7_8)$ because the minimum free distances are 5 and 6 for \mathbf{P}_1 and \mathbf{P}_2 . So we have \mathbf{P}_2 as the puncturing matrix for $(133_8, 171_8)$ convolutional codes.

6.2.1.2 Puncturing design on rate 3/4 codes



The puncturing matrix of a rate-3/4 code punctured by a rate-1/2 code is as

$$\mathbf{P} = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \end{pmatrix}$$

and has 5 different choices of the puncturing matrices

$$\mathbf{P}_1 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad \mathbf{P}_2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad \mathbf{P}_3 = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

$$\mathbf{P}_4 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \quad \mathbf{P}_5 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

In Table 6-3 and 6-4, we have the idea to choose the puncturing matrices for rate-3/4 puncturing convolutional codes. The minimum free distance of the punctured $(5_8, 7_8)$ convolutional code is three, and the code with puncturing matrix \mathbf{P}_1 has the

minimum c_3 . Hence \mathbf{P}_1 is the puncturing matrix of the rate-3/4 $(5_8, 7_8)$ convolutional codes. The same analysis can be applied to $(133_8, 171_8)$ codes and we choose \mathbf{P}_1 as the puncturing matrix.

Code polynomial	R	puncturing matrix	$a_d, d=1,2,\dots,\dots\dots$
5 7	3/4	\mathbf{P}_1	0, 0, 6, 23, 78, 214, ...
		\mathbf{P}_2	0, 1, 6, 21, 70, 191, ...
		\mathbf{P}_3	0, 0, 6, 23, 78, 214, ...
		\mathbf{P}_4	0, 2, 8, 23, 72, 187, ...
		\mathbf{P}_5	0, 0, 6, 42, 74, 182, ...
133 171	3/4	\mathbf{P}_1	0, 0, 0, 0, 7, 20, 56, 118, ...
		\mathbf{P}_2	0, 0, 0, 3, 0, 39, 0, 253, ...
		\mathbf{P}_3	0, 0, 0, 1, 10, 19, 60, 112, ...
		\mathbf{P}_4	0, 0, 0, 4, 11, 21, 71, 116, ...
		\mathbf{P}_5	0, 0, 0, 1, 9, 21, 53, 125, ...

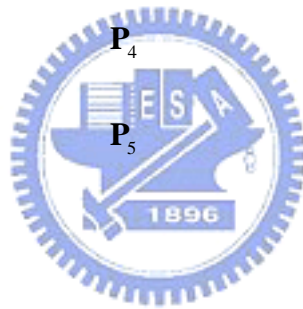


TABLE 6-3: The number of the incorrect paths for the rate-3/4 punctured

$(5_8, 7_8)$ and $(133_8, 171_8)$ codes.

Code polynomial		R	puncturing matrix	$c_d, d=1,2,\dots$
5	7	3/4	\mathbf{P}_1	0, 0, 29, 163, 662, 1949, ...
			\mathbf{P}_2	0, 4, 25, 152, 597, 1984, ...
			\mathbf{P}_3	0, 0, 30, 175, 706, 2028, ...
			\mathbf{P}_4	0, 10, 44, 179, 625, 1751, ...
			\mathbf{P}_5	0, 0, 32, 321, 701, 1871, ...
133	171	3/4	\mathbf{P}_1	0, 0, 0, 0, 38, 90, 294, 669, ...
			\mathbf{P}_2	0, 0, 0, 20, 0, 193, 0, 1420, ...
			\mathbf{P}_3	0, 0, 0, 3, 57, 77, 325, 621, ...
			\mathbf{P}_4	0, 0, 0, 15, 57, 101, 397, 629, ...
			\mathbf{P}_5	0, 0, 0, 6, 38, 105, 290, 664, ...

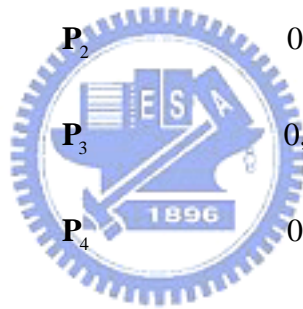


TABLE 6-4: The total number of the error bits produced by the incorrect paths for the rate-3/4

punctured $(5_8, 7_8)$ and $(133_8, 171_8)$ codes

6.2.2 Puncturing patterns design with EXIT chart


The codes with punctured by different puncturing matrices will produce codes with different structures and then the corresponding EXIT charts will be different.

Based on this observation, we can design puncturing matrices with EXIT chart.

6.2.2.1 Rate 2/3 codes

There are still two puncturing matrices \mathbf{P}_1 and \mathbf{P}_2 for the rate-2/3 punctured codes as equation (6.5). The EXIT chart analyses are depicted as follows.

6.2.2.1.1 $(5_8, 7_8)$ convolutional codes



In Fig.6-8, it shows that punctured convolutional codes with different puncturing matrices have different transfer characteristics. The code with matrix \mathbf{P}_2 has the intersection with the detector at the point A where the mutual information of the decoder output is about 0.7 bits. But the code with matrix \mathbf{P}_1 intersects the detector transfer curve at the point B where the mutual information of the decoder output is almost 1. Theoretically, the larger the output mutual information of the decoder is, the better the performance will be. Hence, we choose the puncturing matrix \mathbf{P}_1 when we want to puncture the $(5_8, 7_8)$ convolutional code.

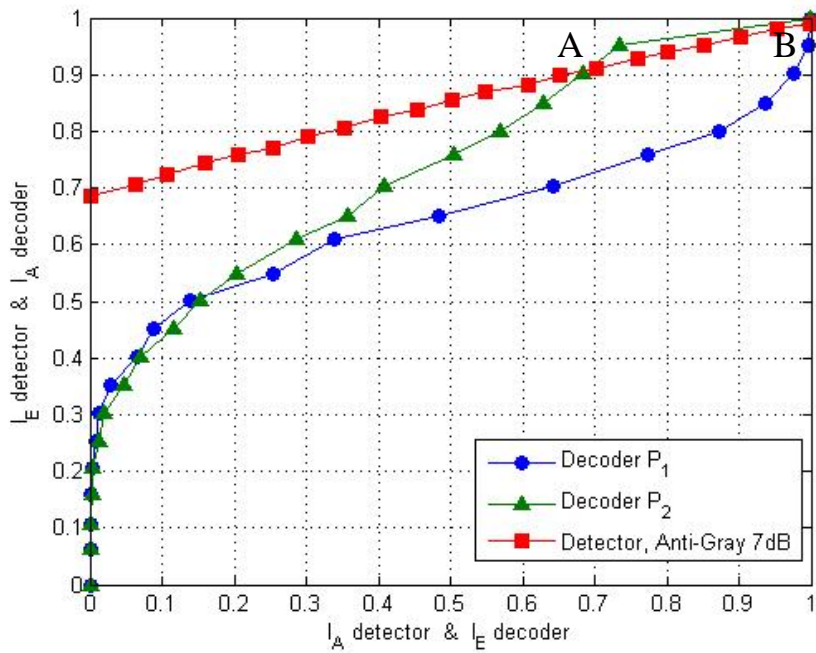


Fig. 6-8: EXIT chart of the punctured codes with puncturing matrices \mathbf{P}_1 and \mathbf{P}_2 at $E_b/N_0 = 7dB$ with

16QAM AntiGray mapping and generator polynomial $(g_0, g_1) = (5_8, 7_8)$.



6.2.2.1.2 $(133_8, 171_8)$ convolutional codes

Fig.6-9 shows that output mutual information I_E of the decoder with puncturing matrix \mathbf{P}_2 when interest with the transfer curve of the detector is smaller than that of the decoder with puncturing matrix \mathbf{P}_1 . So we choose \mathbf{P}_2 when we puncture the $(133_8, 171_8)$ convolutional code to rate-2/3.

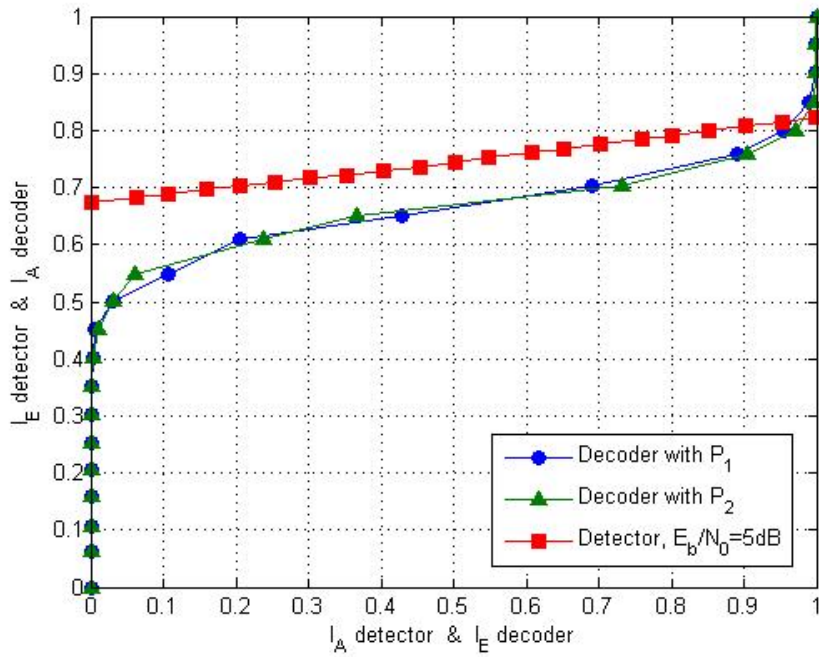


Fig. 6-9: EXIT chart of the punctured codes with puncturing matrices \mathbf{P}_1 and \mathbf{P}_2 at $E_b/N_0 = 5dB$ with

16QAM AntiGray mapping and generator polynomial $(g_0, g_1) = (133_8, 171_8)$



6.2.2.2 Rate 3/4 codes

6.2.2.2.1 $(5_8, 7_8)$ convolutional codes

Similar design criteria will be applied here. In Fig.6-10, we show the transfer characteristics of the decoder with puncturing matrices from \mathbf{P}_1 to \mathbf{P}_5 and that of the detector at $E_b/N_0 = 8dB$ using 16QAM and Natural mapping. In Fig.6-11 we can observe visibly that the intersection point of the decoder using puncturing matrix \mathbf{P}_1 with the detector has the largest output decoder extrinsic mutual

information. So we choose \mathbf{P}_1 as the puncturing matrix when we want to puncture a

$(g_0, g_1) = (5_8, 7_8)$ convolutional code to rate-3/4.

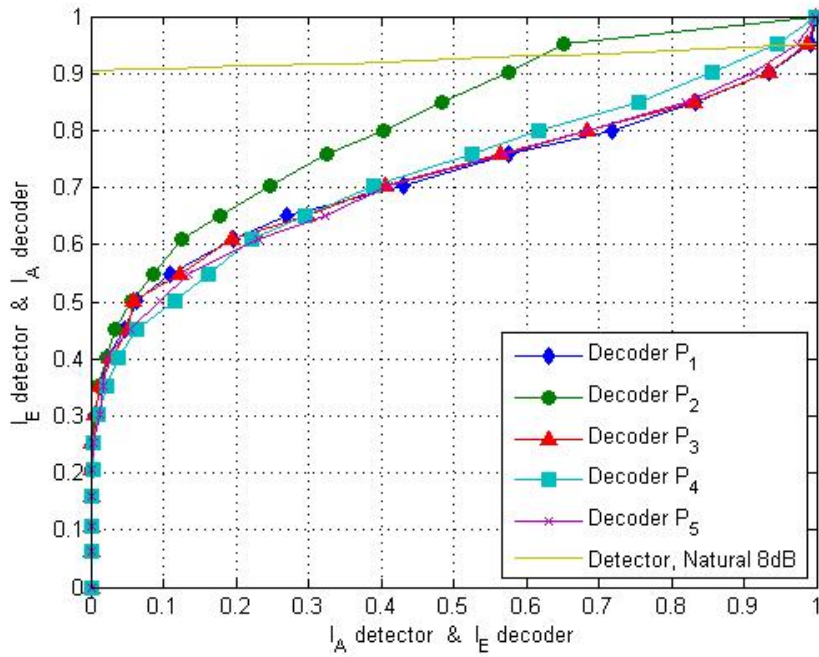


Fig. 6-10: EXIT chart of the punctured codes with puncturing matrices \mathbf{P}_1 to \mathbf{P}_5 at

$E_b / N_0 = 8dB$ with 16QAM Natural mapping and generator polynomial $(g_0, g_1) = (5_8, 7_8)$.

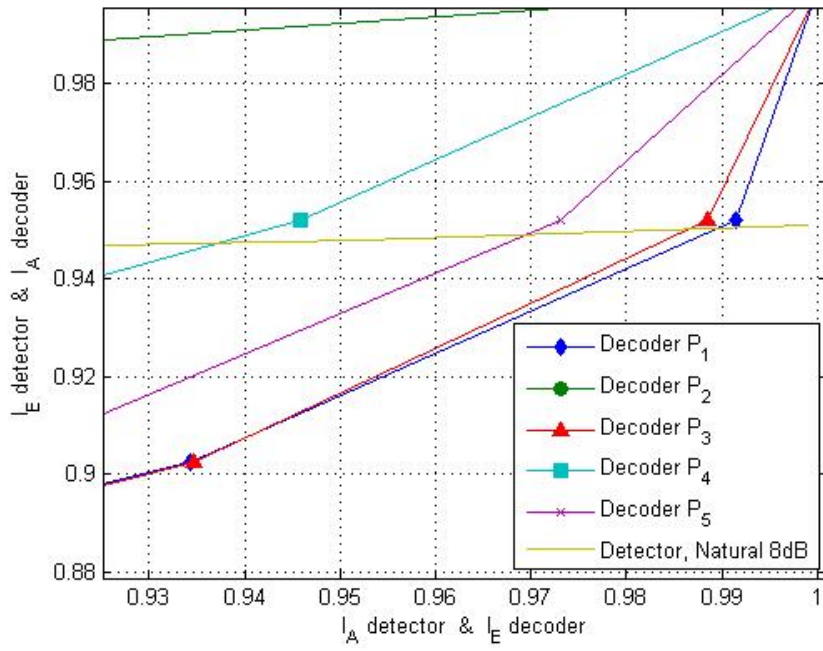


Fig. 6-11: The same chart as Fig.5-3, but the range is from about 0.9 to 1.

6.2.2.2.2 $(133_8, 171_8)$ convolutional codes



In Fig.6-12 and 6-13, we also can have the same observations as in section 6.2.2.2.1. P_1 is still the best choice for rate-3/4 punctured $(133_8, 171_8)$ convolutional codes.

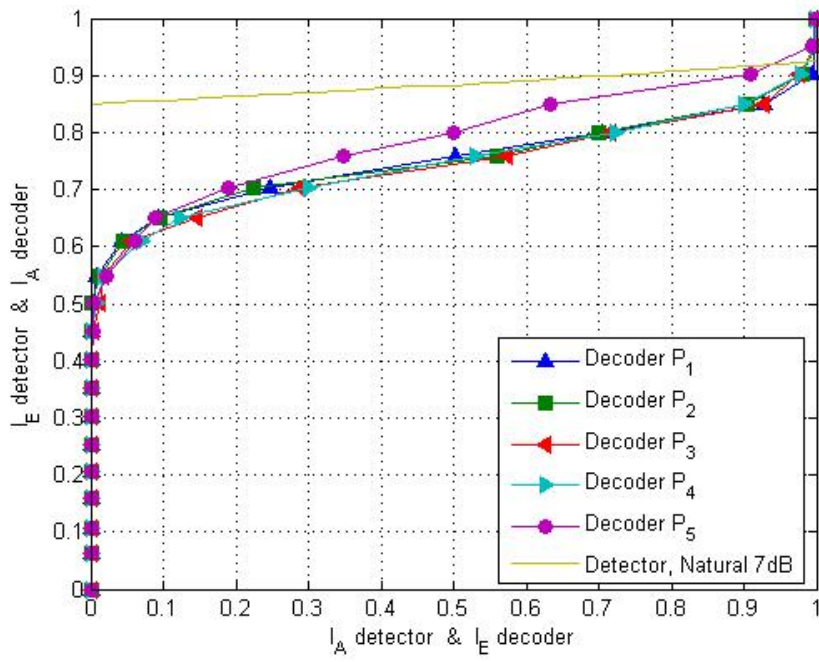


Fig. 6-12: EXIT chart of the punctured codes with puncturing matrices \mathbf{P}_1 to \mathbf{P}_5 at

$E_b / N_0 = 7dB$ with 16QAM Natural mapping and generator polynomial $(g_0, g_1) = (133_8, 171_8)$.

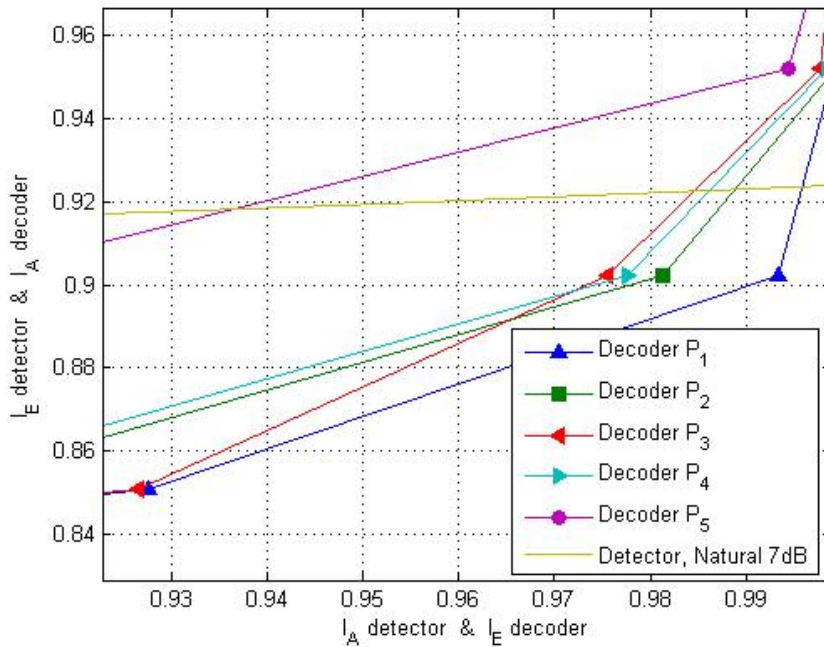
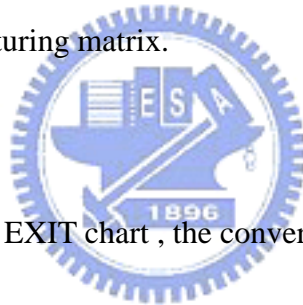


Fig. 6-13: The same chart as Fig.5-5, but the range is from about 0.9 to 1.

6.2.3 Summary on the puncturing design

It is a coincidence that the examples in section 6.2.2 all have the same choices. Other codes in [12] , like $(23_8, 25_8)$ and $(53_8, 57_8)$, choose the puncturing matrices \mathbf{P}_4 and \mathbf{P}_5 respectively. From the simulations of section 6.2.1 and 6.2.2 , the puncturing patterns design will have the same results whether we use the criteria of minimum free distance or the EXIT chart simulations. Once we have decided the code polynomial and the puncturing rate, we can just run the EXIT chart simulations and get the choice of the puncturing matrix.



Besides the simple of the EXIT chart , the convergence behavior can also be seen from it. These advantages will be very useful for us to design a system.