Kasetsart J. (Nat. Sci.) 46 : 305 - 317 (2012)

On Adjusting Vector Symbol Decoding for Many Different Nonbinary Convolutional Codes

Usana Tuntoolavest* and Chanothai Chaiwan

ABSTRACT

The use of nonbinary convolutional codes with Vector Symbol Decoding (VSD) was proposed to correct long burst errors in high data rate wireless networks. VSD allows the symbol size to easily and arbitrarily increase as desired. Therefore, the same technique can be used for both the current data rate and any upcoming data rate of new wireless network standards. Convolutional VSD is more attractive than block VSD due to its lower complexity. Tables of convolutional codes with rate 2/3 and 3/4 are explicitly shown in the feedforward forms as required by VSD. The decoding performance was analyzed for several codes in an additive white Gaussian noise (AWGN) and a Rayleigh fading channel. The performance of VSD for different levels of decoder complexity was also tested. From the results, three design options that provide good performance were discussed including their advantages and disadvantages in terms of flexibility, complexity and ease of implementation.

Keywords: burst error correcting, nonbinary convolutional codes, Vector Symbol Decoding (VSD), error correcting codes

INTRODUCTION

In the near future, the data rate of wireless networks will increase substantially since new wireless standards require a big increase in their data rates. For example, the data rate of the fourth generation mobile (4G) jumps from 2 megabits per second (Mbps) in the third generation (3G) to 100 Mbps for high mobility users and 1 gigabit per second (Gbps) for low mobility users (Parkvall and Astely, 2009). The wireless network standards are similar. The upgraded Wimax standard in 2011 is expected to have a data rate of 1 Gbps for stationary users (Kaur *et al.*, 2012). IEEE 802.11 WiFi standards (IEEE 802.11ac)will change to 1 Gbps also (Park, 2011). The upcoming trend is to provide users with a device designed for accessing the Internet only, with all programs and data stored on the Internet using cloud computing technique. This trend can be seen in the Chrome operating system (OS) for laptops by Google (Adee, 2010). It is expected that users will demand very high speed, reliable data communications for applications such as retrieving important stored data from the Internet, Internet banking and electronic wallets (Google, 2012). Since wireless channels suffer both random and burst errors, high reliability transmission requires powerful error correcting codes.

Almost all error correcting codes in use are binary codes except for the Reed-Solomon (RS) codes. Most codes are designed to correct random errors, not burst errors. To use them in wireless channels, interleaving is often used (Lin

Department of Electrical Engineering, Faculty of Engineering, Kasetsart University, Bangkok 10900, Thailand.
* Corresponding author, e-mail: fengunt@ku.ac.th

Received date : 11/10/11 Accepted date : 02/03/12

and Costello, 2004). The larger the depth of the interleaver is, the longer the delays will be since the decoder must receive the whole code word before it can decode (Vanichchanunt *et al.*, 2009). For nonbinary codes, correcting a symbol is equal to correcting all error bits in that symbol. Thus, they are suitable for correcting burst errors in the wireless channel.

One of the most widely used nonbinary codes is the (255,223) RS code with 8-bit symbols due to its byte-size symbols and its high code rate. However, its decoding algorithm is complex. It can correct up to 16 error symbols or fill in 32 erasure symbols (Lin and Costello, 2004), which are enough for a moderate data rate but will not be sufficient for future data rates. For example, increasing the data rate from 10 Mbps to 1 Gbps means that the number of bits in the error burst also increases 100 times because the burst duration remains the same. Although interleaved RS (IRS) codes or corroborative IRS codes may be used to increase the symbol size as shown in Justesen *et al.* (2004) and Schmidt *et al.* (2009), it is not suitable at this scale.

For the 3G standard, Turbo codes and LDPC (low density parity check) codes are used (Papaharalabos *et al.*, 2011). Dai and Cai (2010) proposed that LDPC codes with a long code length had better performance than Turbo codes and claimed that it would be an important technology for future high speed wireless communication. Details of Turbo codes can be found in Sripimanwat (2005). LDPC is a binary code and can be used with another nonbinary code in a concatenated code structure.

The aim of this paper was to propose the use of nonbinary convolutional codes in a concatenated coding scheme to correct very long burst errors. The Vector Symbol Decoding (VSD) algorithm for convolutional codes proposed by Tuntoolavest and Metzner (2002) allows the symbol size to easily and arbitrarily increase as necessary. It can be easily adjusted for the increase in the data rate. Convolutional VSD is attractive because it can start decoding after it has received only a few symbols (Tuntoolavest and Metzner, 2002). This reduces the complexity and the required memory of the decoder. By nature, convolutional codes use more redundancy than block codes (Lin and Costello, 2004). This becomes a much smaller problem for high data rates because compared to entertainment data such as a movie or video-on-demand, text-based data that requires high reliability and powerful codes is usually much smaller in size. This is because human perception is not perfect and can accept some errors in images or movies. However, errors in a text file will probably lead to misinterpretation. Thus, with the added redundancy, the important data can still be transmitted in a very short time.

For this paper, the VSD algorithm was modified from the original algorithm (Tuntoolavest and Metzner, 2002) to work with different code structures. Several convolutional codes were also expressed with rate 2/3 and 3/4 in the feedforward form required by VSD. The modifications and the code structure in the feedforward form are necessary for the performance analysis of codes with different rates and constraint lengths in AWGN and the Rayleigh fading channels. The performance for different levels of decoder complexities was tested and three coding design options were considered in terms of flexibility, complexity and ease of implementation.

BACKGROUND

Vector symbol decoding

VSD is a decoding technique that is specifically suitable for linear codes with large nonbinary symbols (Tuntoolavest, 2004). This makes it suitable for correcting burst errors. VSD can be applied to block codes and convolutional codes. This paper focuses on VSD for convolutional codes. The first step of the VSD algorithm is to compute a syndrome (S_1) after receiving a subblock of symbols. The number of symbols in a subblock depends on the code rate as shown in Figure 1. If this syndrome is an all-zero vector, the decoder decides that there is no error in this particular subblock. If the syndrome is not an all-zero vector, the decoder will consider an additional subblock of symbols, compute the next syndrome value (S_2) and attempt to make corrections. The error positions can be found by the error-locating vector. Then, the error values can be computed from a matrix inversion.

VSD can be used with list decoding (Metzner, 2003), which assumes that the inner decoder can provide a list of two alternate choices for each received symbol. When the first choice is wrong and the second choice is correct, VSD can easily recognize this and replace the wrong first choice by the correct second choice in most cases. This helps reduce the number of error symbols that the main VSD algorithm called as “correct with null combination” has to handle. It also helps reduce the overall complexity of the outer decoder. In a normal concatenated coding system, the inner decoder is not a list decoder. Thus, the outer decoder receives only one possible value for each received symbol. In this case, VSD without list is used and always attempts to correct the errors using the null combinations. The overall complexity of the outer decoder is higher than the one that employs VSD with list. However, the main advantage is that other inner decoders can be used instead of the list Viterbi algorithm.

List Viberti algorithm-vector symbol decoding concatenated coding system

The list Viberti algorithm-vector symbol decoding (LVA-VSD) system shown in Figure 2 is a generalized concatenated coding system in the sense that the inner and the outer code can be any combination of block and/or convolutional codes. The encoding part is straightforward and simple. The nonbinary encoder is based on the same circuit as the binary one. The block cyclic encoder can use a similar shift register circuit to the RS encoder (Lin and Costello, 2004). The nonbinary convolutional encoder is also based on the structure of the binary encoder. The difference is that the exclusive-OR (XOR) operations are performed on the symbol basis and each memory unit must store a symbol instead of 1 bit. This was explained in detail by Tuntoolavest and Intharasakul (2006).

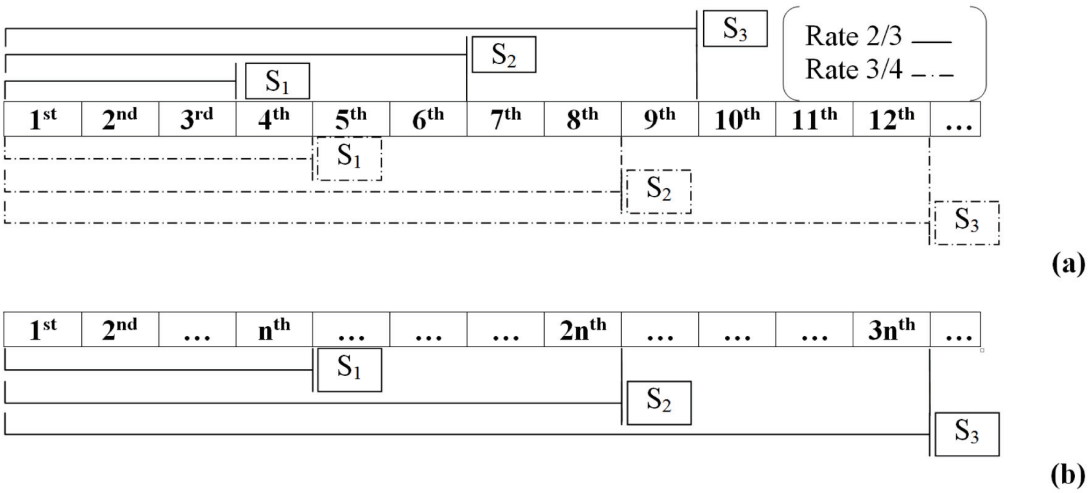

Figure 1 Number of symbols required for computing each syndrome (S) for: (a) rate 2/3, rate 3/4; (b) rate (n-1)/n convolutional codes.

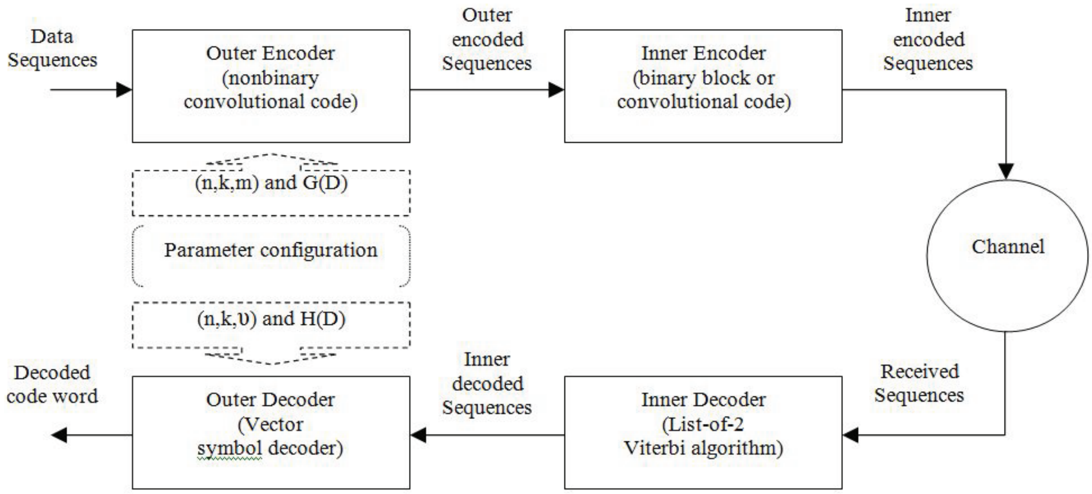

Figure 2 Block diagram of list Viberbti algorithm-vector symbol decoding (LVA-VSD) concatenated coding system.

The decoding part is more complex. VSD was used as the outer decoder. The list-of-2 Viterbi (LVA) was chosen as the inner decoder. The list-of-L Viterbi was proposed by Seshadri and Sundberg (1994) to provide L possible decoded sequences in the order of their likelihood for each received sequence. For the LVA-VSD system, list-of-2 is optimum in terms of performance and complexity (Tuntoolavest and Seubnaung, 2007). LVA for block codes was also demonstrated in (Tuntoolavest *et al.*, 2011).

THE PROPOSED METHOD

The more flexible vector symbol decoding algorithm

For this paper, the VSD algorithm used was modified from the algorithm by (Tuntoolavest and Metzner, 2002) and the additional step by Tuntoolavest (2007) to allow different code structures. The original algorithm was hard-coded for a particular (3,2,2) convolutional outer code and many parameters in the decoding steps were fixed for that structure. It was also necessary to add two more functions to convert the generator matrix and the parity check matrix.

These modifications make VSD more flexible in terms of code rates and constraint lengths. Specifically, it allows any rate 2/3 and rate 3/4 outer convolutional codes. It is then possible to compare the performance of several codes under different conditions. Consequently, appropriate designed parameters can be found for each application.

With the modified VSD, the user needs to input the generator transfer matrix **G(D)**, the parity transfer matrix **H(D)** and the *n, k, m, υ* values of the selected code into the program, where *n* is the number of output symbols for each set of *k* input symbols, *m* is the memory size and *υ* is the overall constraint length. The values of all these parameters are described explicitly in the next subsection. After receiving these parameters, the VSD program will then generate the outer encoder and the outer decoder functions for the selected code.

The outer encoder

For the encoder part, the steps to construct the shift register circuit are:

1. The user inputs parameter *n*, *k* and *m* for a *k/n* convolutional code with *m* time delay and also inputs generator sequences in octal.

2. The program converts generator vectors $g_{i,j}$ from octal to binary sequences. Then, the shift register circuit is generated for the selected *k/n* convolutional code.

Example 1: For a selected (4,3,2) convolutional code with an overall constraint length of 4, the steps are:

1. The user inputs $n = 4$, $k = 3$ and $m = 2$

$g_{1,1} = 0_8$, $g_{1,2} = 1_8$, $g_{1,3} = 2_8$, $g_{1,4} = 3_8$

$g_{2,1} = 3_8$, $g_{2,2} = 0_8$, $g_{2,3} = 1_8$, $g_{2,4} = 2_8$

$g_{3,1} = 2_8$, $g_{3,2} = 4_8$, $g_{3,3} = 1_8$, $g_{3,4} = 5_8$

2. The program converts each input generator sequence to a binary sequence.

For example, $g_{1,1} = 000_2$, $g_{1,2} = 001_2$, $g_{1,3} = 010_2$, $g_{1,4} = 011_2$

It also generates the encoding circuits. Specifically, each output sequence v_i is based on the structure shown in Figure 3.

The parity check matrix

Normally the decoding of binary convolutional codes does not require knowledge of the parity check matrix since they are usually decoded with the Viterbi algorithm (Lin and Costello, 2004) and a trellis diagram. However, the Viterbi algorithm is not suitable for decoding convolutional codes with large nonbinary symbols because it would require a large and impractical number of states. VSD achieves the ability to decode large nonbinary convolutional codes without using this large number of states by observing that a terminated convolutional code is similar to a linear block code in some ways. Specifically, a syndrome can be computed from the multiplication of a received sequence and the parity check matrix. Thus, the VSD algorithm requires the parity check matrix of the selected code as an input.

In the previous design (Tuntoolavest and Metzner, 2002), the decoder was used for a particular rate 2/3 code. To investigate the performance of the higher rate code and to make the decoder more flexible, the modified VSD allows other rate 2/3 and rate 3/4 convolutional codes. These codes were selected from the optimum codes tabulated in Chang *et al.* (1997) and are shown in the next section. The steps to convert the codes in Chang *et al.* (1997) into the parity check matrix format for VSD are:

1. The user inputs parameter *n*, *k* and υ for a *k/n* convolutional code with υ overall constraint length and also inputs parity check sequences in octal.

2. The program converts parity check vectors h_i from octal to binary sequences. Then, the sub-parity check matrix is generated.

3. The sub-parity check matrix is used for generating the parity check matrix for the selected *k/n* convolutional code.

Example 2: For a selected (4,3,2) convolutional code with an overall constraint length of 4, the steps are:

1. The user inputs $n = 4$, $k = 3$ and $\upsilon = 4$

$h_3 = 33_8$, $h_2 = 25_8$, $h_1 = 37_8$, $h_0 = 31_8$

2. The program converts the input parity check sequences to

$h_3 = 11011_2$, $h_2 = 10101_2$, $h_1 = 11111_2$, $h_0 = 11001_2$

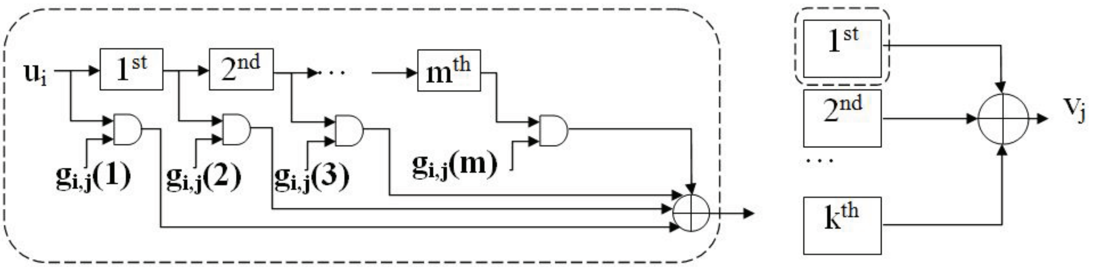

Figure 3 Each encoded sequence (v_i) of an (n, k, m) convolutional code.

Since each $\boldsymbol{h_i}$ is the column vector of the sub matrix **H,** the sub parity check matrix is

$$\mathbf{H_{sub}} = [h_3 \ h_2 \ h_1 \ h_0] = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad [1]$$

3. The program arranges the sub-parity check matrix to the corresponding semi-infinite **H** matrix similar to that of a binary convolutional code. Each sub-matrix in box is repeated as shown in Equation 2 with the dashed line arrow. The **H** matrix for this particular code is:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & \cdots \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \cdots \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & \cdots \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad [2]$$

4. The parity check matrix will be used for computing syndromes and correcting the errors in the wrong received sequences.

Nonbinary convolutional codes

For convolutional VSD, the outer codes should be non-systematic convolutional codes with a relatively high rate. The non-systematic structure is chosen because the generator sequences of good codes with large free distance (d_{free}) are mostly non-systematic in the feedforward form. The feedforward form is preferred because the all-zero sequence can be used to terminate the code word. This periodic termination helps limit the propagation of errors to the next code word. It also increases the probability of correct decoding since the values of the tail symbols (all-zero) are known. The disadvantage is that it adds more redundancy to the code word. The relatively high rate is used to reduce some redundancy and increase the data rate. It should also be noted that by nature, convolutional codes usually require more redundancy than block codes. However, they also have a simpler encoding and decoding mechanism (Lin and Costello, 2004).

Table 1 explicitly shows the pairs of generator sequences and parity check sequences that are in the form required by VSD. These sequences were adapted from the feedback encoder proposed by Chang *et al.* (1997) and converted to the feedforward form using the techniques described in Porath (1989). This table is useful because the usual tabulated good non-systematic convolutional codes for rate 2/3 and 3/4 in texts such as Lin and Costello (2004) are defined by their parity check sequences only. In addition, the suggested method in Lin and Costello (2004) converted the given parity check sequences to the corresponding generator sequences in the feedback form, not the feedforward form.

Performance investigation approaches

Two approaches were used to find the performance at the outer code layer. First, assume the input symbol error probability. Second, simulate the inner code layer for each channel condition. The decoding failure probability of the inner decoder is the input symbol error probability of the outer decoder. The first approach is useful because the results are independent from the types of inner codes and inner decoders. The second approach is necessary for the LVA-VSD system. Since list-of-2 VA is a list decoding algorithm, it produces a list of two possible decoded sequences. The error probability of the most likely and the second most likely decoded sequences are correlated and must be known for accurate simulations of VSD. For the simulations, five different outer convolutional codes were employed. These codes were the (3,2,2) codes with the overall constraint length of 3 and 4 and the (4,3,2) codes with the overall constraint length

Table 1 Table of 2/3 and 3/4 convolutional codes.

Code rate	m	υ	G(D)	H(D)
2/3	1	2	$\begin{bmatrix}3&1&0\\2&3&3\end{bmatrix}$	$\begin{bmatrix}3&5&7\end{bmatrix}$
	2	3	$\begin{bmatrix}3&2&1\\4&1&7\end{bmatrix}$	$\begin{bmatrix}17&15&13\end{bmatrix}$
		4	$\begin{bmatrix}6&5&1\\7&2&5\end{bmatrix}$	$\begin{bmatrix}23&31&27\end{bmatrix}$
	3	5	$\begin{bmatrix}07&06&03\\12&01&13\end{bmatrix}$	$\begin{bmatrix}71&57&73\end{bmatrix}$
		6	$\begin{bmatrix}06&13&13\\13&06&17\end{bmatrix}$	$\begin{bmatrix}123&147&121\end{bmatrix}$
	4	7	$\begin{bmatrix}16&13&0\\25&05&34\end{bmatrix}$	$\begin{bmatrix}313&27&241\end{bmatrix}$
		8	$\begin{bmatrix}37&31&16\\2&14&35\end{bmatrix}$	$\begin{bmatrix}555&631&477\end{bmatrix}$
	5	9	$\begin{bmatrix}27&23&16\\46&17&41\end{bmatrix}$	$\begin{bmatrix}1051&1423&1327\end{bmatrix}$
		10	$\begin{bmatrix}53&51&34\\52&37&55\end{bmatrix}$	$\begin{bmatrix}2621&2137&3013\end{bmatrix}$
3/4	1	2	$\begin{bmatrix}1&1&1&0\\3&0&0&1\\3&2&0&2\end{bmatrix}$	$\begin{bmatrix}2&5&7&6\end{bmatrix}$
		3	$\begin{bmatrix}3&2&1&0\\3&1&2&1\\2&2&2&3\end{bmatrix}$	$\begin{bmatrix}11&13&15&12\end{bmatrix}$
	2	4	$\begin{bmatrix}0&1&2&3\\3&0&1&2\\2&4&1&5\end{bmatrix}$	$\begin{bmatrix}33&25&37&31\end{bmatrix}$
		5	$\begin{bmatrix}3&3&2&2\\5&2&7&0\\4&7&0&1\end{bmatrix}$	$\begin{bmatrix}47&73&57&75\end{bmatrix}$
		6	$\begin{bmatrix}5&4&3&2\\4&6&5&5\\6&1&4&3\end{bmatrix}$	$\begin{bmatrix}107&135&133&141\end{bmatrix}$
	3	7	$\begin{bmatrix}02&03&04&07\\03&07&03&05\\15&02&02&17\end{bmatrix}$	$\begin{bmatrix}211&341&315&267\end{bmatrix}$
		8	$\begin{bmatrix}04&06&07&07\\01&12&05&14\\00&07&14&11\end{bmatrix}$	$\begin{bmatrix}535&757&733&661\end{bmatrix}$
		9	$\begin{bmatrix}03&06&10&15\\00&16&03&13\\16&05&02&17\end{bmatrix}$	$\begin{bmatrix}1475&1723&1157&1371\end{bmatrix}$

m = memory size; *υ* = overall constraint length; **G(D)** = (transfer domain) generator matrix; **H(D) =** (transfer domain) parity matrix.

of 4, 5 and 6. Their structures are shown in Table 1. Two cases of VSD were also investigated. The first one was the basic VSD without list decoding and the second one was the VSD with list-of-2 decoding. Each outer code word contained 60 32-bit symbols for all cases.

The level of VSD complexity can be limited by the number of syndromes allowed in the calculation. It was known that increasing the number of syndromes leads to better decoding capability for the same code. In this paper, this effect was investigated further to demonstrate how it affected codes with different rates and constraint lengths. To obtain this information, VSD without list was simulated with three different maximum numbers of syndromes—namely, 8, 12 and 16—being the highest number of syndromes that the decoder was allowed to use in each case. This number is directly related to the complexity and the memory size of the decoder which are particularly important when the algorithm is implemented in an electronic board. For VSD with list, the maximum number of syndromes was set only 12 to because the second choices help reduce the required number of syndromes. The effect of the list was investigated for different code rates and constraint lengths.

RESULTS

The results are divided into two subsections which follow the first and second approaches explained in the method section, respectively.

Effect of constraint length and level of complexity

Figure 4 shows that the limit on the decoder complexity has an interesting effect on the performance of VSD. Normally, the code with the higher constraint length (υ) is expected to perform better than the code with the lower constraint length. However, if the number of syndromes allowed was too few, the lower constraint length could produce the better performance because the lower constraint length means that each symbol is related to fewer symbols in the past. As seen from Figure 4, the decoding failure probability of the (4,3,2) code with υ = 6 is substantially better for the 16-syndrome case than the 8 and 12-syndrome cases, which do not differ much. For the case with υ = 5, this probability is noticeably different for all three syndrome cases and it is better when more syndromes are allowed, as expected. For the case with v = 4, the 12 and 16 syndromes produce

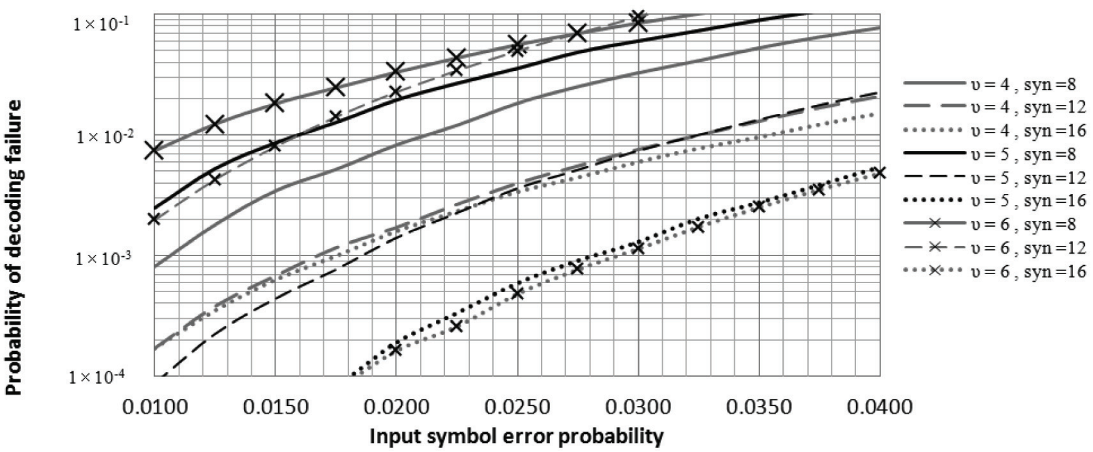

Figure 4 Performance of Vector Symbol Decoding for three (4,3,2) convolutional codes with different constraint length (υ) and different number of maximum syndromes (syn).

similar results, but they are significantly better than the 8-syndrome case.

Considering the trade-off between the complexity and performance, Figure 4 shows that if 16 syndromes are allowed, the code with the constraint length of 5 is optimum. If no more than 12 syndromes are allowed, the code with the constraint length of 4 is optimum.

Figure 5 compares the performance of a rate 2/3 and a rate 3/4 code with the same constraint length of 4 and shows that the lower rate code provides a much lower decoding failure probability when the same number of syndromes are allowed. In addition, increasing the complexity (number of syndromes) has much less effect than reducing the rate.

Figure 6 reveals the percentage that the decoder uses for each level of syndrome numbers.

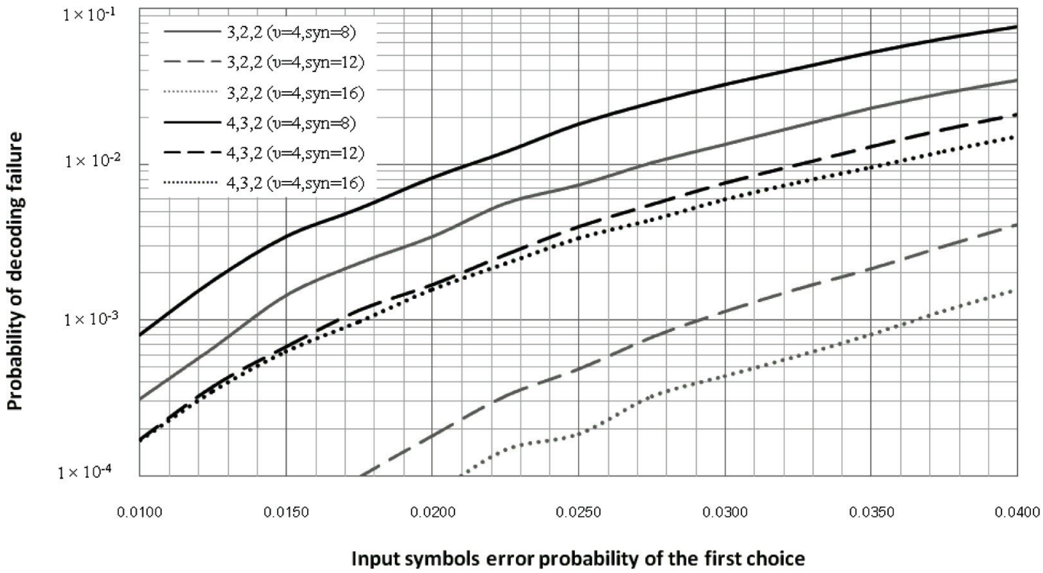

Figure 5 Performance of Vector Symbol Decoding for a (3,2,2) and a (4,3,2) and convolutional code with the same constraint length at different number of maximum syndromes (syn).

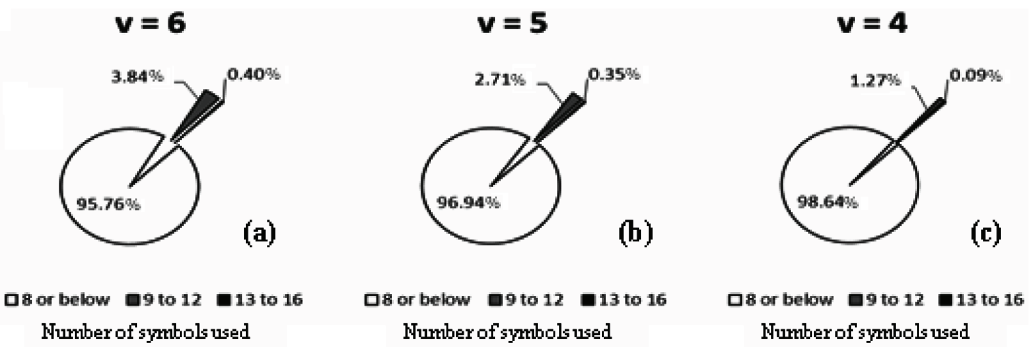

Figure 6 Percentage of number of syndromes used in decoding the (4,3,2) convolutional code in an additive white Gaussian noise (AWGN) channel at E_b/N_0 = 2.3 dB with different constraint length (υ): (a) υ = 6; (b) υ = 5; (c) υ = 4.

The channel condition is the AWGN channel with E_b/N_0 = 2.3 dB. For more than 95% of the time, the decoder uses no more than 8 syndromes. The higher the constraint length, the higher percentage the decoder uses for the 9 to 12 and 13 to 16 syndromes. This confirms that for a high constraint length code, it is necessary to allow a large number of syndromes.

Effect of second choices

Another factor that affects the performance of VSD is the use of second choices. If the inner decoder can provide two alternative choices for each decoded inner sequence, VSD can use this extra information to improve its performance and reduce its complexity. The symbol error probabilities for the first and second choices were from decoding a (2,1,4) convolutional inner code with LVA in the AWGN channel at E_b/N_0 = 2.3 dB and in the Rayleigh fading channel.

Figure 7 shows that when the second choice is available, the decoder can decode with fewer syndromes. This is true for both the rate 2/3 and rate 3/4 codes and true for both low and high constraint lengths. This means the decoder for the rate 2/3 code is much less complex than the one for the rate 3/4 code because the number of calculations increases exponentially with the number of syndromes. Therefore, the availability of the second choice allows the use of a smaller number of syndromes and the lower rate code is preferred in terms of complexity.

Figures 8 and 9 show the simulation results for VSD with list-of-2 in AWGN and the Rayleigh fading channel for four different codes. The fading channel is assumed to be independently

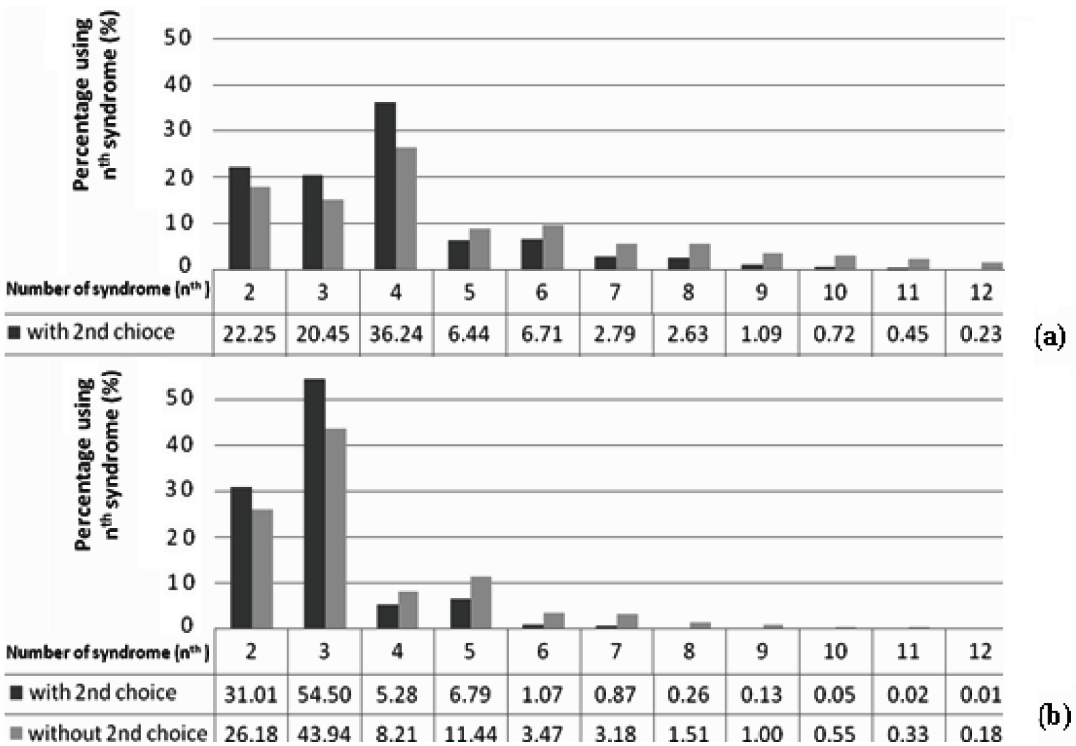

Figure 7 Histogram of percentage use of the number of syndromes in an additive white Gaussian noise (AWGN) channel at E_b/N_0 = 2.3 dB in decoding of: (a) a (4,3,2) convolutional code with υ = 5; (b) a (3,2,2) convolutional code with υ = 3.

fading according to a Rayleigh distribution for each bit. This channel may be justified by interleaving a long sequence of bit. The channel is also corrupted by additive white Gaussian noise. The modulation scheme for this channel is frequency shift keying and square law detection is used at the receiver. The performances of each code are in the same order in both channels. The (3,2,2) code with υ = 4 is the best. The (4,3,2) code with υ = 4 is the worst. The results also show that increasing the constraint length improves the lower rate code more than the higher rate code.

DISCUSSION

The results clearly show that the constraint length affects the optimum maximum number of syndromes. For a relatively short constraint length, a large number of syndromes is not necessary and the decoder is suitable with

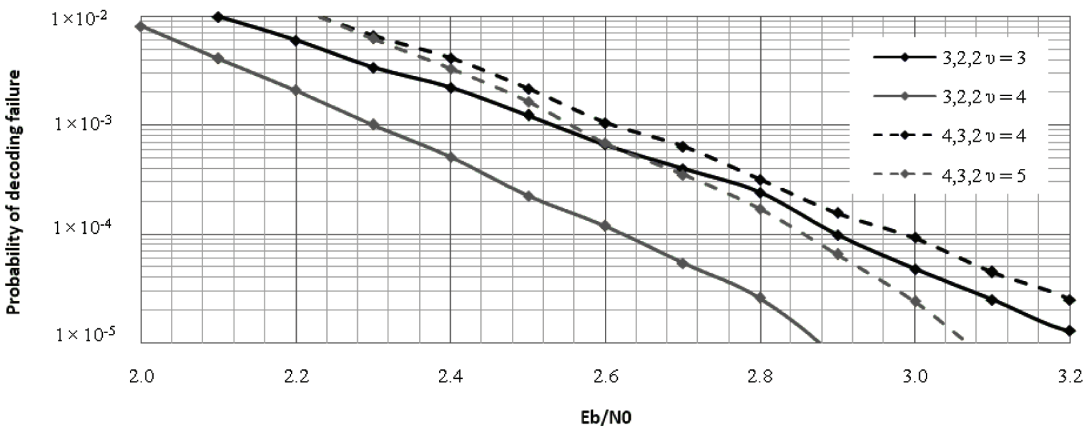

Figure 8 Performance of Vector Symbol Decoding with list-of-2 inner decoder for four different convolutional codes in an additive white Gaussian noise (AWGN) channel.

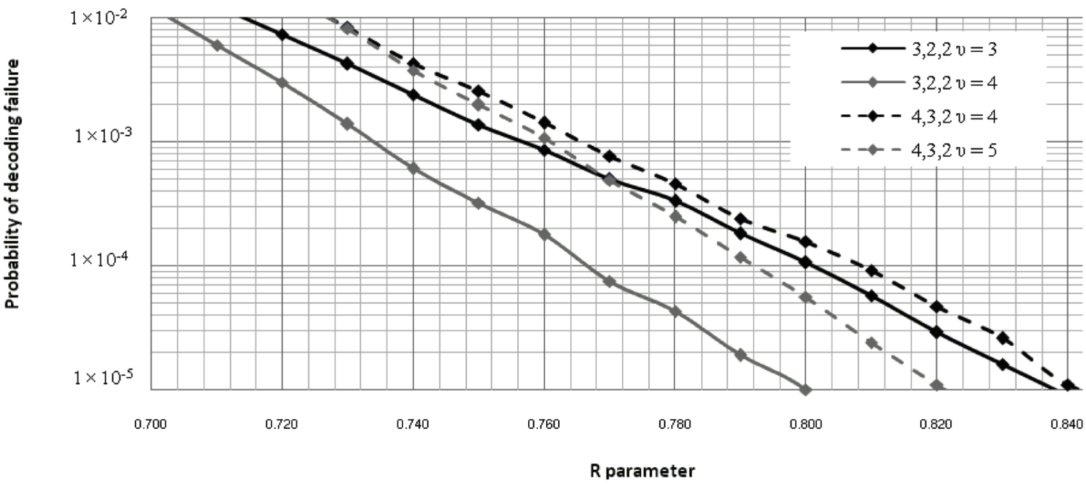

Figure 9 Performance of Vector Symbol Decoding with list-of-2 inner decoder for four different convolutional codes in a Raleigh fading channel.

fewer syndromes. For a relatively long constraint length, it is necessary to allow a large number of syndromes. If a large number of syndromes is not possible due to hardware limitations, a lower constraint length code is preferable. In addition, if the inner decoder can provide alternative choices, VSD will be able to decode with fewer syndromes. Therefore, the options in the design of this coding system are:

Option 1: Use a list inner decoder and a small number of syndromes for VSD.

Advantages: VSD can decode quickly with low complexity and is easy to implement.

Disadvantages: The inner decoder is not very flexible and is more complex. It is troublesome to apply LVA for block codes. However, space diversity such as multiple antennas or time diversity may be used to provide alternative choices instead of LVA.

Option 2: Use any inner decoder without a list and with larger number of syndromes for VSD.

Advantages: The inner decoder is very flexible. The existing decoder can be used. There is no need for a new implementation.

Disadvantages: VSD is more complex and is harder to implement.

Option 3: Use any inner decoder without a list and with small number of syndromes for VSD with a low rate and low constraint length convolutional code.

Advantages: The inner decoder is very flexible as in option 2. VSD can decode quickly with limited complexity and is easier to implement than option 2.

Disadvantages: The code rate is low, so there is a lot of redundancy.

Other techniques may use interleaved RS (IRS) codes or corroborative IRS codes (Justesen *et al.*, 2004; Schmidt *et al.*, 2009) to handle burst errors with large symbols. Such techniques have several differences to the proposed technique. First, they are mainly suitable for block outer codes. Second, list inner decoding is not used since the outer decoder was not designed for it. Third, the suitable symbol size is typically shorter than the one in the proposed technique.

CONCLUSION

Nonbinary convolutional codes in a concatenated coding scheme with a VSD outer decoder can be a solution for burst error correcting in very high speed wireless networks. In principle, VSD can be applied with any large symbol size. This makes it attractive for the upcoming standard with a much higher data rate. However, its performance will greatly deteriorate if there are many random errors in addition to the burst errors. Therefore, it should be used with a good inner code that can correct those random errors. Since the increase in symbol size will affect the design of the inner code, the effective symbol size is subject to the length of good inner codes. In addition to a wireless fading channel, VSD might be applied for a channel with impulsive noise such as the power line channel because the impulsive noise also results in burst errors.

ACKNOWLEDGEMENTS

This research was funded by the Faculty of Engineering, Kasetsart University.

LITERATURE CITED

Adee, S. 2010. Chrome the conqueror. **IEEE Spectrum** 47(1): 34–39.

Chang, J., D. Hwang and M. Lin. 1997. Some extended results on the search for good convolutional codes. **IEEE Trans. Inf. Theory** 43(5): 1682–1697.

Dai, N. and L.Cai. 2010. LDPC Encoding Based on the Primitive Polynomial, pp. 2991–3012. *In* **Wireless Comm. Network and Mobile Comp. (WiCOM),** 23–25 September 2010. Chengdu, China.

Google. 2012. Google wallet. [Available from http://www.google.com/wallet/how-it-works.html]. [Sourced: 2 April 2012].

Justesen, J., C. Thommesen and T. Høholdt. 2004. Decoding of concatenated codes with interleaved outer codes, pp.329. *In* **IEEE Int. Symp. Inf. Theory (ISIT)**, 27 June–2 July 2004. Chicago, USA.

Kaur, M.J., M. Uddin, H.K. Verma, B.R. Ambedkar. 2012. Role of cognitive radio on 4G communications a review. **J. of Emerg. Trends in Comput. and Infor. Sciences** 3(2): 272–276.

Lin, S. and D.J. Costello Jr. 2004. **Error Control Coding,** 2nd ed. Pearson Education, Upper Saddle River, NJ, USA. 1,260 pp.

Metzner, J.J. 2003. Vector symbol decoding with list inner symbol decision. **IEEE Trans. on Comm.** 51(3): 371–380.

Papaharalabos, S., D. Benmayor, P.T. Mathiopoulos and P. Fan. 2011. Performance comparisons and improvements of channel coding techniques for digital satellite broadcasting to mobile users. **IEEE Trans. on Broadcast.** 57(1): 94–102.

Parkvall, S. and D. Astely. 2009. The evolution of LTE towards LTE advanced. **J. of Comm.** 4 (3): 146–154.

Park, M. 2011. IEEE 802.11ac: Dynamic bandwidth channel access, pp. 1–5. *In* **IEEE ICC 2011**, 5–9 June 2011. Kyoto, Japan.

Porath, J. 1989. Algorithms for converting convolutional codes from feedback to feedforward form and vice versa. **Electron. Lett.** 25(15): 1008–1009.

Schmidt, G., V.R. Sidorenko and M. Bossert. 2009. Collaborative decoding of interleaved Reed–Solomon codes and concatenated code designs. **IEEE Trans. on Inf. Theory** 55(7): 2991–3012.

Seshadri, N. and C.W. Sundberg. 1994. List Viterbi decoding algorithms with applications. **IEEE Trans. on Comm.** 42: 313–323.

Sripimanwat, K. 2005. **Turbo Code Applications: A Journey from a Paper to Realization.** Springer, 386 pp.

Tuntooolavest, U. 2004. A simple method to improve the performance of convolutional vector symbol decoding with small symbol size, pp. 676–679. *In* **IEEE TENCON 2004**, 21–24 November 2004. Chiang Mai, Thailand.

Tuntoolavest, U. 2007. Additional steps of convolutional vector symbol decoding for general data sequence, pp. 651–654. *In* **The 2007 ECTI International Conference,** 9-12 May 2007. Chang Rai, Thailand.

Tuntoolavest, U. and R. Intharasakul. 2006. Nonbinary convolutional encoder for vector symbol decoding on FPGA board, pp. 716–719. *In* **The 10th Int. Conf. on Comm. Tech. (ICCT2006),** 27–30 November 2006. Guilin, China.

Tuntoolavest, U. and J.J. Metzner. 2002. Vector symbol decoding with list symbol decisions and outer convolutional codes for reliable communications. **Integr. Computer-aided Engineering** 9(2): 101–116.

Tuntoolavest, U. and A. Seubnaung. 2007. Performance investigation of convolutional vector symbol decoding with larger than two choices and with incomplete second choices. **Kasetsart J. (Nat. Sci.)** 41: 364–370.

Tuntoolavest, U., V. Suktalordcheep and C. Chaiwan. 2011. List-of-2 soft decision viterbi inner decoder for a generalized concatenated coding system, pp. 264–267. *In* **ECTI-CON 2011,** 17-19 May 2011. Khon Kaen, Thailand.

Vanichchanunt, P., P. Kovintavewat, U. Tuntoolavest, K. Sripimanwat, K. Woradit, P. Sangwongngam, D. Chumchewkul and L. Wuttisittikulkij. 2009. **Channel Coding Theory** (in Thai). TRIDI, 392 pp.