PAPER Special Section on Information Theory and Its Applications

Iterative Decoding of High Dimensionality Parity Code

Toshio FUKUTA[†], ***Student Member***, **Yuuichi HAMASUNA**[†], **Ichi TAKUMI**[†a)], **Masayasu HATA**[††], ***Regular Members***, ***and*** **Takahiro NAKANISHI**[†††], ***Nonmember***

SUMMARY Given the importance of the traffic on modern communication networks, advanced error correction methods are needed to overcome the changes expected in channel quality. Conventional countermeasures that use high dimensionality parity codes often fail to provide sufficient error correction capability. We propose a parity code with high dimensionality that is iteratively decoded. It provides better error correcting capability than conventional decoding methods. The proposal uses the steepest descent method to increase code bit reliability and the coherency between parities and code bits gradually. Furthermore, the quantization of the decoding algorithm is discussed. It is found that decoding with quantization can keep the error correcting capability high.

key words: *high dimensionality parity code, iterative decoding, steepest descent method, bit reliability, quantization*

1. Introduction

We have been researching the high dimensionality parity code (HDPC). Its structure is so simple that hardware implementation can be realized easily [1]. Additionally, high decoding speeds are also expected [2], [3]. This code can deal with not only random errors but also burst errors through the addition of our original interleaving function [4] which diffuses errors within a codeword. Conventional decoding methods that are based on high dimensionality parity code include "Iterative Corrections on every 2D sub-code" [5] and "Majority Logic Decoding" [1]. They involve hard decision decoding and iteration of deterministic error corrections. If the bit error rate(BER) is around 10^{-3}, conventional decoding methods offer adequate error correcting capability. At BERs under 10^{-1}–10^{-2}, however, the conventional methods fail to provide sufficient error correction capability.

Our proposal, the iterative decoding with steepest descent method, offers high error correcting ability because the value of each code bit is assigned a reliability term [7]. This paper explains the features and the decoding algorithm with steepest descent method. A comparison of its error correcting capability to that of a conventional decoding method confirms its superiority. Since iterative calculation of code bit reliability is excessively expensive if floating point calculations are used, we consider the quantization of the decoding algorithm. Although quantization reduces computational cost, it is possible that error correcting capability may be degraded. In our computer simulations, this algorithm with limited accuracy of the high dimensionality parity code keeps high error correcting capability.

Manuscript received January 20, 2003.
Manuscript revised April 19, 2003.
Final manuscript received June 6, 2003.

[†]The authors are with the Department of A.I. and Computer Science, Nagoya Institute of Technology, Nagoya-shi, 466-8555 Japan.
[††]The author is with Chubu University, Kasugai-shi, 487-8501 Japan.
[†††]The author is with Gunma University, Maebashi-shi, 371-8510 Japan.
a) E-mail: takumi@ics.nitech.ac.jp

2. High Dimensionality Parity Code

2.1 Construction of HDPC

HDPC is an extension of a 2 dimensional direct product code of single parity codes to higher dimensions. It has an n dimensional discrete hyper-cubical structure. Each edge is m bits long, and the code block consists of m^n code bits. Code structure is denoted as nDmm, when n is dimension and m is size. For example, 3Dm4 means a code with dimension of three and size of four as shown in Fig. 1. nDmm code has $(m-1)^n$ information digits and m^n total digits including parity redundant digits, so the transmission rate is $R = (1 - 1/m)^n$.

2.2 Transmission Order of Code Bit

The transmission order of HDPC is set to isolate the transmission intervals between the bits on the same parity check line as described in Ref. [6]. The minimum difference in code bit transmission order on the same par-

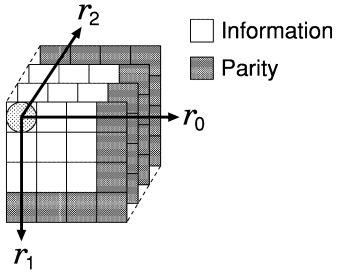

Fig. 1 Parity check for each bit (3Dm4 code).

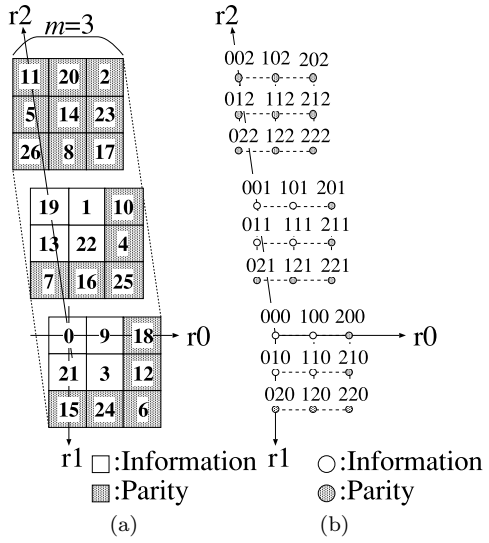

Fig. 2 Relation between the transmission order of 3Dm3 code and coordinates (r_0, r_1, r_2).

ity check line is described in the appendix. This transmission order ensures that burst errors are distributed uniformly within a code. An example of the transmission order for the 3Dm3 code is shown in Fig. 2(a). The numbers attached to the bits indicate the transmission order. Figure 2(b) shows the position of each bit using coordinates. When this coordinate expression is used, the bit transmitted as the t-th$(0 \leq t \leq m^n - 1)$ bit has coordinates $(r_0, r_1, \cdots, r_{n-1})$ given by the following formulas.

$$\left.\begin{array}{rcl} r_0 & = & \left(\displaystyle\sum_{i=0}^{n-1} \lfloor t/m^i \rfloor\right) \bmod m \\ r_k & = & \lfloor t/m^{n-(k+1)} \rfloor \bmod m \\ & & (k = 1, 2, \cdots, n-1) \end{array}\right\} \quad (1)$$

3. Conventional Decoding Methods

This section briefly reviews the conventional decoding methods of Iterative Correction on every 2D sub-code and Adaptive Threshold Decoding.

3.1 Iterative Correction on Every 2D Sub-Code

The behavior of the conventional decoding algorithm of Iterative Correction on every 2D sub-code is very easy to understand. The nDmm code has m 2Dmm sub-codes which are parallel in the code structure. Since the nDmm code has n axes, it has $m^{n-2} \cdot {}_n\mathrm{C}_2$ 2Dmm sub-codes. Each 2Dmm sub-code can correct a single bit error. If a sub-code is challenged with multiple errors, the error correction is abandoned to other sub-codes which contain a part of the error bits. This algorithm does not fully utilize the code dimension of n.

1. Investigate e_i for each code bit $C_i (0 \leq i \leq m^n - 1)$. e_i is the number of parity check lines penetrating C_i and on which parity error has been detected.
2. $e_{\max}$ is the maximum of all e_i.
   a. When $e_{\max}$ is 0: go to 4.
   b. When $e_{\max}$ is not 0: Bit C_i which has e_i equaling $e_{\max}$ is reversed.
3. 1.–2. is repeated a predetermined number of times (Hard Decision Limit).
4. Decoding completed.

Fig. 3 Adaptive threshold decoding algorithm.

3.2 Adaptive Threshold Decoding

Each code bit in nDmm is checked by n orthogonal parity check lines. When a majority of n parity check lines penetrating a code bit detect error, the code bit is flagged for correction. Majority logic decoding offers higher error correction performance than Iterative Correction on every 2D sub-code, but Adaptive Threshold Decoding described in Fig. 3 is the best of all hard decision algorithms proposed for high dimensionality parity code, because it has the fewest erroneous corrections in which correct bits are incorrectly flagged.

4. Iterative Decoding with Steepest Descent Method

4.1 The Correcting Method

4.1.1 Bit Reliability α_i

We start by defining the bit reliability, α_i, of each bit. α_i expresses the probability that the bit is right $(0 \leq \alpha_i \leq 1)$. When the bit is presumed correct by parity check, α_i should be increased. On the other hand, when the bit is presumed wrong, α_i should be decreased. Error bits are specified by iterating these presumptions. This paper assumes that basically no bit is known to be corrupted before error decision, that is, all bits are hardly decided before error correction, so all α_i values are set equal in the initial condition.

4.1.2 The Reliability β_j of Parity Check

The next step is to define the reliability, β_j, of each parity check. Each parity check is reliable with probability $\beta_j (0 \leq \beta_j \leq 1)$; this represents the consistency between parity check result and all bit reliabilities α_i. A nDmm code has m bits on every parity check line. The reliabilities of m bits on the line are denoted by

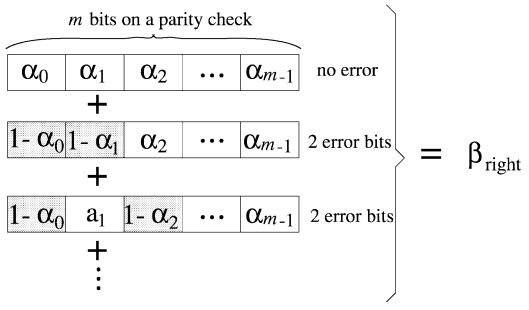

Fig. 4 Calculation of β_{right} for a parity check. The gray box means error bit.

$\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{m-1}$.

Describing the notation more properly, we can denote j by coordinates $(r_0, r_1, .., r_{k-1}, x, r_{k+1}, .., r_{n-1})$ when the parity line lies in parallel with the r_k axis. (x has no meaning other than indicating the position of k.) The parity line penetrates the m code bits whose coordinates are $(r_0, r_1, .., r_{k-1}, i, r_{k+1}, .., r_{n-1})$ $(i = 0, 1, .., m-1)$. α_i is the reliability of a code bit at that coordinate.

The reliability of the parity check is β_{right} if the parity check has not detected an error. β_{right} is probability of any even numbers of error bits occuring on the parity line (Fig. 4). β_{right} is given by

$$\begin{aligned}\beta_{\text{right}} &= \alpha_0\alpha_1\alpha_2\cdots\alpha_{m-1} \\ &\quad + (1-\alpha_0)(1-\alpha_1)\alpha_2\cdots\alpha_{m-1} \\ &\quad + (1-\alpha_0)\alpha_1(1-\alpha_2)\cdots\alpha_{m-1} + \cdots\cdots \\ &= \frac{1}{2} + \frac{1}{2}\prod_{k=0}^{m-1}(2\alpha_k - 1). \end{aligned} \tag{2}$$

β_{right} has a value between 0 and 1, because all α_k $(k = 0, 1, .., m-1)$ lie between 0 and 1.

We say that the reliability of the parity check is β_{wrong} if the parity check has detected an error. β_{wrong} is probability of any odd number of error bits occuring on the parity line. β_{wrong} also equals to $1 - \beta_{\text{right}}$ and given by

$$\beta_{\text{wrong}} = \frac{1}{2} - \frac{1}{2}\prod_{k=0}^{m-1}(2\alpha_k - 1). \tag{3}$$

β_{wrong} has a value between 0 and 1 for the same reason as β_{right}.

Therefore, reliability β_j of each parity check is β_{right} when the real parity check detects no error, and β_j is β_{wrong} when the real parity check detects an error.

When no bits or an even number of code bits on a parity line have reliabilities α_i under 0.5, the corresponding β_{right} will be greater than 0.5. If the real parity check does not detect parity error, bit reliability α_i and the real parity check are consistent, and α_i should be modified to increase β_{right}; that is, to increase β_j. If the real parity check does detect parity error, bit reliability α_i and the real parity check are inconsistent, and α_i should be modified to increase β_{wrong}, i.e. increase β_j.

Similarly, when an odd number of code bits on a parity line have reliabilities α_i under 0.5, consistency between α_i and real parity check is achieved by modifying α_i to increase β_j.

Here, $\beta_{\min}$ is the minimum of all β_j. When $\beta_{\min}$ exceeds 0.5, no error would be detected by any parity check line if the bits whose α_i is lower than 0.5 have been reversed. Therefore, all α_i are modified so that all β_j exceed 0.5. Note that the modification of α_i is stopped and hard decided when all β_j exceed 0.5.

4.1.3 Modification of Bit Reliability α_i

We use the so-called gradient algorithm to increase β_j by modifying α_i. In the algorithm, bit reliabilities α_i are the controlled parameters and parity check reliabilities β_j are the evaluation function or objective function. α_i are increased or decreased according to the partial differential coefficient of β_j. If the step size of the gradient algorithm is small enough, parameters α_i converge at the point where β_j are maximum. If all β_j equal 1 after convergence, parameter α_i must be at the global maximum, which means that the code block has recovered consistency with regard to the parity check.

As for α_i, the partial differential coefficients of β_{right} and β_{wrong} are derived from Eqs. (2) and (3) as follows,

$$\frac{\partial\beta_{\text{right}}}{\partial\alpha_i} = \prod_{k=0,k\neq i}^{m-1}(2\alpha_k - 1) \tag{4}$$

$$\frac{\partial\beta_{\text{wrong}}}{\partial\alpha_i} = -\prod_{k=0,k\neq i}^{m-1}(2\alpha_k - 1), \tag{5}$$

where,

$$0 \leq \frac{\partial\beta_{\text{right}}}{\partial\alpha_i} \leq 1 \tag{6}$$

$$-1 \leq \frac{\partial\beta_{\text{wrong}}}{\partial\alpha_i} \leq 0 \tag{7}$$

because of the range of α_k $(k = 0, 1, .., m-1)$.

One of the two above equations (Eqs. (4) and (5)) is employed as $\partial\beta_j/\partial\alpha_i$ according to the real parity check mentioned above.

Since the steepest descent method is applied to the modification of α_i, α_i should be increased corresponding to $\partial\beta_j/\partial\alpha_i$. In an nD$m$$m$ code, α_i is checked by n parity checks whose reliabilities are $\beta_0, \beta_1, \beta_2, \ldots, \beta_{n-1}$. α_i should be increased corresponding to $\sum_{j=0}^{n-1}\partial\beta_j/\partial\alpha_i$ in order to increase β_j $(j = 0, 1, .., n-1)$ in total. Since modified α_i should lie in the range 0 to 1, the modification formula is given by Eq. (8).

1. All bit reliabilities α_i are set to the same initial value.
2. The reliability β_j of each parity check is calculated by Eqs. (2) and (3).
3. a. When the minimum value $\beta_{\min}$ of β_j is higher than 0.5: go to 5.
   b. When $\beta_{\min}$ is 0.5 or less: the bit reliability, α_i, of each bit is modified by Eq. (8).
4. 2.–3. is repeated until the iteration number reaches the preset limit (Iteration Limit).
5. The bits whose α_i is lower than 0.5 are reversed.
6. Decoding completion.

Fig. 5 Iterative decoding algorithm for high dimensionality parity code.

$$\alpha_i \leftarrow \begin{cases} \alpha_i + \dfrac{1-\alpha_i}{n}\displaystyle\sum_{k=0}^{n-1}\frac{\partial\beta_k}{\partial\alpha_i} & \left(when \ \displaystyle\sum_{k=0}^{n-1}\frac{\partial\beta_k}{\partial\alpha_i} \geq 0\right) \\ \alpha_i + \dfrac{\alpha_i}{n}\displaystyle\sum_{k=0}^{n-1}\frac{\partial\beta_k}{\partial\alpha_i} & \left(when \ \displaystyle\sum_{k=0}^{n-1}\frac{\partial\beta_k}{\partial\alpha_i} < 0\right) \end{cases} \tag{8}$$

where the second terms have the ranges of

$$0 \leq \frac{1-\alpha_i}{n}\sum_{k=0}^{n-1}\frac{\partial\beta_k}{\partial\alpha_i} \leq 1-\alpha \tag{9}$$

$$-\alpha_i \leq \frac{\alpha_i}{n}\sum_{k=0}^{n-1}\frac{\partial\beta_k}{\partial\alpha_i} < 0 \tag{10}$$

because of Eqs. (6), (7) and the cases implied by the summation of Eq. (8). The new α_i will lie between 0 and 1.

The modification of bit reliability α_i of each bit is repeated by iterating Eq. (8). This iteration should ensure that α_i of the right bit converges to 1 while α_i of the error bit converges to 0 so as to achieve better consistency between the parity check and bit reliability. After convergence, bits with low α_i are judged to be corrupted and are corrected by hard decision.

4.1.4 Decoding Algorithm

The proposed decoding algorithm is summarized in Fig. 5.

4.2 Behavior of Bit Reliability α_i and Reliability β_j of Parity Check

Proposed decoding was applied to the 3Dm2 code(code length 8, $R = 0.125$) and the behavior of bit reliability α_i and reliability β_j of parity check were observed.

Figure 7 shows the behavior of bit reliability α_i for the case where there are 3 error bits in the code as shown in Fig. 6(a). Notations for α_i are shown in Fig. 6(b). In addition, the initial values of all α_i were set to 0.625, which equals the rate of right bits in the code block. Figure 7 shows the convergence of all α_i to 0 or 1. This means that error bits can be revealed by repeating the modification of α_i.

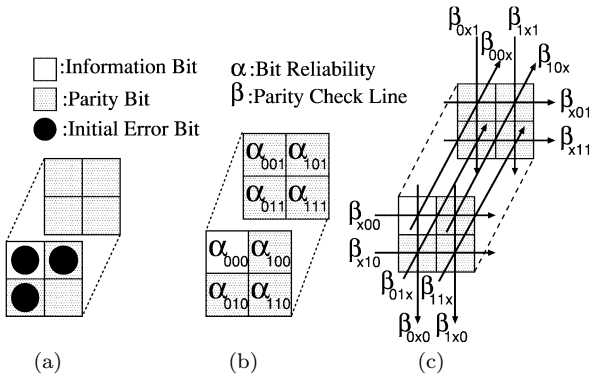

Fig. 6 Initial error bits and notation of α_i and β_j.

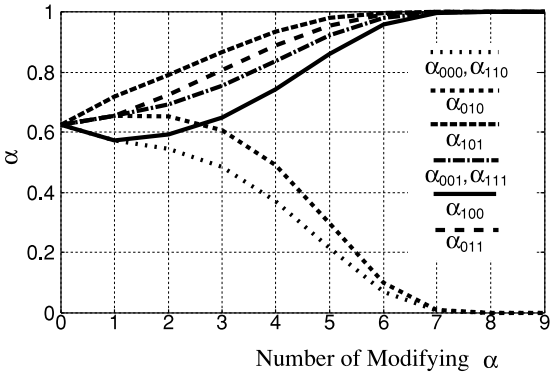

Fig. 7 Behavior of bit reliability α_i.

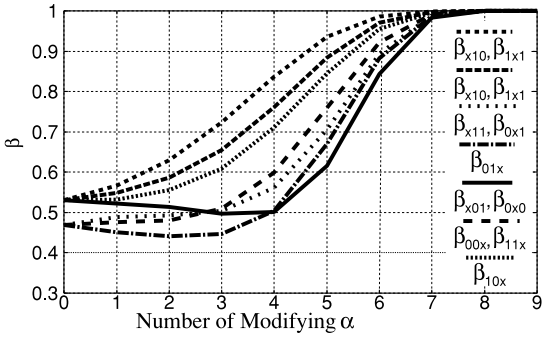

Fig. 8 Behavior of the reliability β_j of parity check.

Figure 8 shows the behavior of the reliability β_j of parity check in this decoding process. Notations for β_i are shown in Fig. 6(c). The value of β_j in the first iteration is divided into two values, 0.531 and 0.469. The first time value of β_j exceeds 0.5 corresponds to the real parity check that does not detect an error. On the other hand, the first time β_j lower than 0.5 corresponds to the real parity check to detect an error. In Fig. 8, all β_j converge to 1 independently, regardless of their

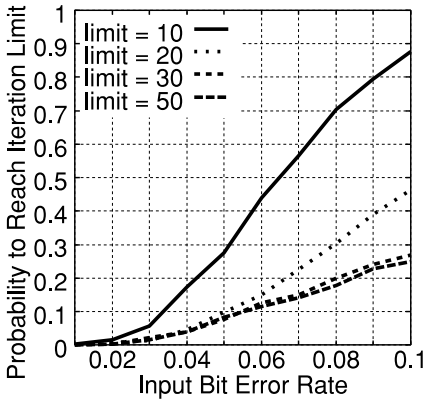

Fig. 9 Probability of iteration reaching the limit (3Dm5, Initial $\alpha_i = 0.8$).

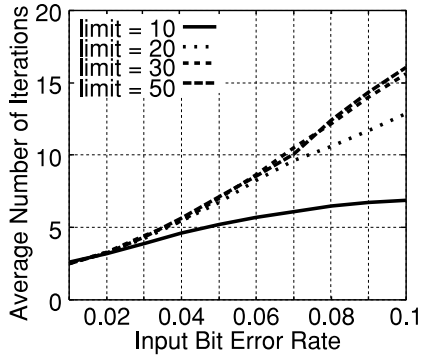

Fig. 10 Average iteration number when iteration stops before the limit (3Dm5, Initial $\alpha_i = 0.8$).

initial value. The convergence of all β_j to 1 means that consistency between all parity checks and all of bit reliabilities has been recovered. So, reversing all bits with $\alpha_i = 0$ makes the code block correct.

4.3 Iteration Limit and Bit Error Rate

Here we discuss the iteration limit and decoded bit error rate (BER) as discerned from the simulation results. In the simulation, initial values of α were 0.8 and 0.85 for 3Dm5 and 4Dm6, respectively. Numbers of samples in the simulation were 10^5 and 10^4, respectively.

Figures 9 and 12 show the probabilities if the iteration number reaching the various iteration limits in 3Dm5 and 4Dm6, respectively. When the iteration number reaches the given limit, some β_j are less than 0.5 and hard decision or error correction based on α_i will leave errors. The probabilities increase according to the input BER as shown in Figs. 9 and 12. Especially in Fig. 12, most iterations reach the limit when the input BER is greater than 0.09; error correcting capability can not be expected at these BER values.

In Fig. 9, the probability of reaching the limit for limit=30 and limit=50 are almost the same. This means that iterations in excess of 50 are useless. On the other hand, Fig. 12 shows that the probabilities for the various limits examined are very similar.

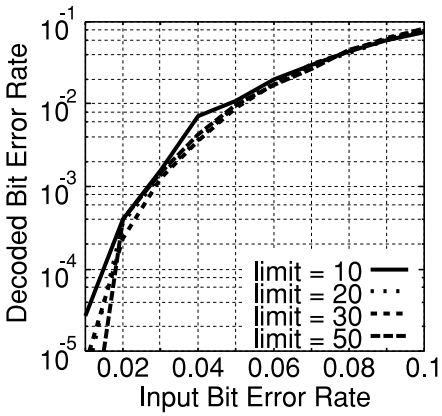

Fig. 11 Decoded bit error rate for various iteration limit values (3Dm5, Initial $\alpha_i = 0.8$).

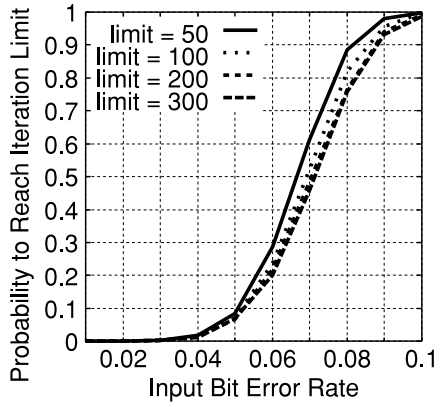

Fig. 12 Probability of iteration reaching the limit (4Dm6, Initial $\alpha_i = 0.85$).

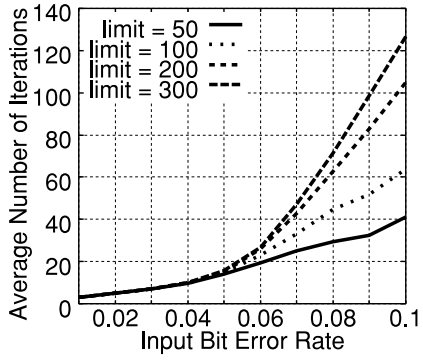

Fig. 13 Average iteration number when iteration stops before the limit (4Dm6, Initial $\alpha_i = 0.85$).

Figures 10 and 13 show the average number of iterations for samples for which the iteration stops before reaching the limit. These averages increase with the input BER, which indicates the difficulty of error correction at high error rates.

Figures 11 and 14 show the decoded BER for all

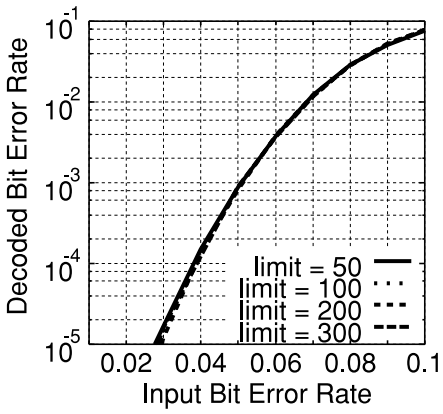

Fig. 14 Decoded bit error rate for various iteration limit values (4Dm6, Initial $\alpha_i = 0.85$).

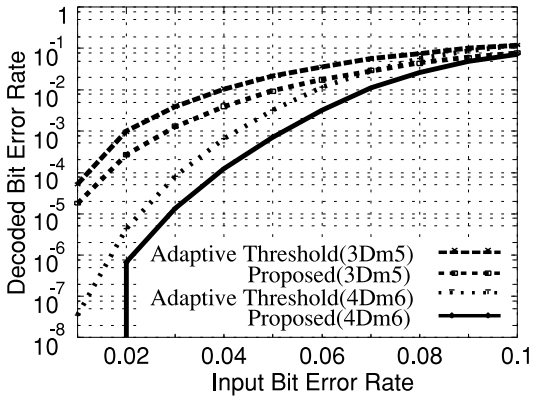

Fig. 15 Comparison of random error correction capability.

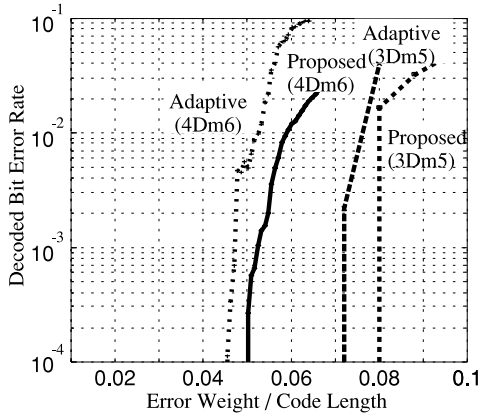

Fig. 16 Comparison if burst error correction capability.

simulation samples. In both figures, the plots are very similar and higher limits slightly decrease the decoded BER at lower input BER. In all cases, the decoded BER does not exceed the input BER.

In practice, the iteration limit is determined by how many iterations are possible in each time period from the moment when a block is received to the moment when the next block is received. Accordingly, the limit is restricted by bit transmission speed and processing speed.

5. Comparison with Conventional Decoding Method

In this section, we compare the proposed decoding method with the conventional decoding method of Adaptive Threshold Decoding in terms of random error correcting capability and burst error correcting capability.

5.1 Random Error Correction

First, random error correcting capability is compared. Figure 15 shows decoded BER for 3Dm5 code ($R = 0.512$) and 4Dm6 code ($R = 0.482$). The hard decision limit, shown in Fig. 3, of Adaptive Threshold Decoding is 15 times, and the bit correcting iteration is fully converged.

On the other hand, the initial value of α_i is 0.80 for the 3Dm5 code with proposed decoding and 0.75 for the 4Dm6 code. The maximum modification number of α_i is 50 times for the 3Dm5 code, and 100 times for the 4Dm6 code. The maximum modification number, that is the iteration limit in the proposed decoding algorithm described in Fig. 5, is discussed in Sect. 4.3, and the given limit of 50 or 100 yields reasonable decoded BER.

Figure 15 shows that the proposed decoding method has higher error correcting capability regardless of code dimensionality or input BER.

5.2 Burst Error Correction

Figure 16 compares the burst error correcting capability of the two methods. In this simulation, each sample of code block contains 1 solid burst error. That is, the BER in the burst equals 1. In practice, however, the BER in a burst generally does not exceed 0.5. Decoding conditions such as codes used, the initial value of α_i, and the maximum modification number of α_i, and so on are the same as used in the above subsection.

Figure 16 shows that the proposed decoding method has higher error correcting capability than Adaptive Threshold Decoding. Furthermore, its error correcting capability is never worse than that of random error correction (see Fig. 15). When the input BER is 0.05 or less and the proposed decoding method is applied, burst errors are corrected completely (Fig. 16). We note the proposed method fails to correct all random errors when the input BER is 0.02 or less as shown in Fig. 15. Overall, this code has higher error correcting capability against burst errors than the conventional code.

6. Quantization of Proposed Algorithm

The proposed decoding method described above is computationally expensive so hardware implementation is difficult. Our solution is to quantize the numerical representations and calculations; i.e. α_i, β_j and differentiated β_j. Figures 17 and 18 show the decoded BER using the quantization procedure; the input BER is 0.05. Figure 17 is for the 3Dm5 code and Fig. 18 is for the 4Dm6 code. The numbers attached to the plots in the figures represent the quantization bit numbers. The error correcting capability of the proposed method without quantization and that of the conventional method are also shown for reference.

Figure 17 shows that 6 bit-quantization yields virtually the same performance as the original code if the initial value of α_i is greater than 0.8. For the 3Dm5 code, therefore, the proposed algorithm can be quantized to 6 bits. Moreover, we see that the proposed method has better error correcting capability than hard decision decoding. Quantization with fewer than 6 bits degrades the error correcting performance because calculated modified values are less than the quantization step size. If the modified value of α_i had been enlarged by the algorithm, performance would have been improved. Some study on formula (Eq. (8)) which modifies α_i, is required

Figure 18 also shows the benefit of 6 bit-quantization if the initial value of α_i lies between 0.85 to 0.95. The impact of the initial value of α_i on error correcting capability must be investigated further to calculate the optimum initial value.

Consider the decoded BER for input BER in the range 0.01 to 0.1 with various quantization values as shown in Figs. 19 and 20, which correspond to 3Dm5 and 4Dm6. Initial α_i values are 0.8 and 0.85, which agree with the above discussion. Once again 6 bit-precision is shown to be sufficient for the proposed method.

Next we consider the circuit scale of the decoders with and without 6 bit-quantization. The above simulation results without quantization were obtained by using 32 bit floating-point arithmetic. However, all values of α_i, β_j and their partial differential coefficients lie between 0 and 1. This means that the practical precision of 32 bit floating-point arithmetic is 24 bit fixed-point: sign bit and 23 bit mantissa. Multiplica-

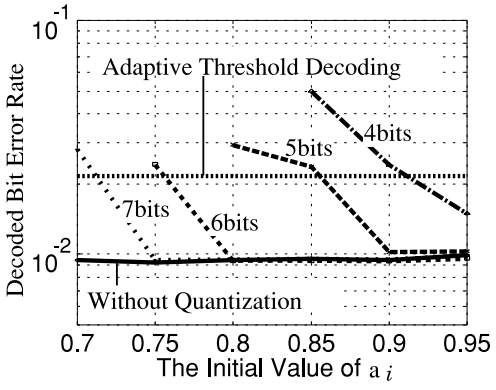

Fig. 17 Decoded BER with quantized algorithm for 3Dm5.

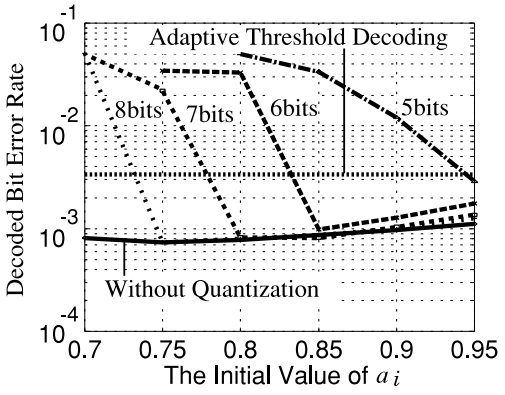

Fig. 18 Decoded BER with quantized algorithm for 4Dm6.

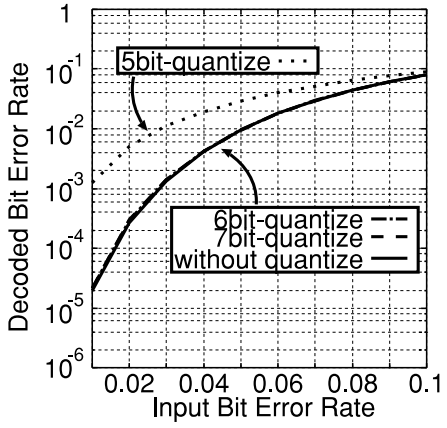

Fig. 19 Decoded BER with quantized algorithm for 3Dm5 (Initial $\alpha_i = 0.8$).

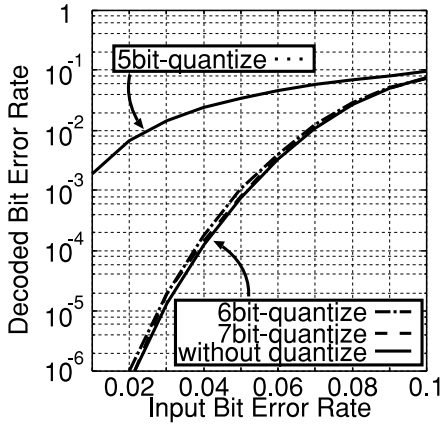

Fig. 20 Decoded BER with quantized algorithm for 4Dm6 (Initial $\alpha_i = 0.85$).

tion and division, present in the proposed algorithm, have circuit scale of $O(\text{word-width}^2)$. Adders and registers correspond to $O(\text{word-width})$. When a data-flow structure is adopted for the decoder, the latter factor is dominant so the circuit scale of the decoder with 6 bit-quantization is at least 75% smaller than that with floating-point.

7. Conclusion

This paper has introduced iterative decoding with steepest descent method for error correction in high dimensionality parity code. Although it remains to fully assess the proposed decoding method, we have delineated its possible performance. We showed that the proposed decoding method was superior to the conventional method in terms of error correcting capability and is feasible. To enhance the ease of implementation, we quantized the iterative decoding algorithm. The proposed decoding algorithm with 6 bit-quantization offers virtually unchanged error correcting capability. What remains to be confirmed is the true lower limit of quantization bit number. To that end, we need to examine the initial value of α_i and refine the modification formula.

References

[1] Y. Hamasuna, M. Yamamura, T. Ishizaka, M. Matsuo, M. Hata, and I. Takumi, "Hardware implementation of the high-dimensional discrete torus knot code," IEICE Trans. Fundamentals, vol.E84-A, no.4, pp.949–956, April 2001.
[2] S.I.R. Costa, E. Agustini, M. Muniz, and R. Palazzo, Jr., "Slepian-type codes on a flat torus," ISIT2000, p.58, June 2000.
[3] D. Rankin and T.A. Gulliver, "Serial and parallel concatenation of SPC product codes," ISITA2000, W-B-1-2, pp.778–781, Nov. 2000.
[4] M. Hata, E. Yamaguchi, and I. Takumi, "Probalistic error correcting code based on mean BER—Design of high-dimensional discrete torus knot code," Proc. SITA'98, pp.169–172, Dec. 1998.
[5] S. Kuroda, E. Yamaguchi, I. Takumi, and M. Hata, "A geometrical decoding algorithm and correcting limit of high-dimensional hyper-cubic-ring code—Correcting ability in BER of 10^{-1}–10^{-2}," IEICE Trans. Fundamentals (Japanese Edition), vol.J80-A, no.12, pp.2145–2154, Dec. 1997.
[6] K. Hashimoto and M. Hata, "High-dimensional symmetric parity check code capable of correcting 10^{-1}–10^{-2} random errors," IEICE Trans. Fundamentals (Japanese Edition), vol.J75-A, no.8, pp.1257–1266, Aug. 1992.
[7] T. Fukuta, T. Usuda, I. Takumi, and M. Hata, "Soft decision decoding of a high dimensionality parity code," IEICE Technical Report, IT2001-8, May 2001.

Appendix: The Minimum Difference of Transmission Order on the Same Parity Check Line

Equation (1) is shown again as follows.

$$r_0 = \left(\sum_{i=0}^{n-1} \lfloor t/m^i \rfloor\right) \bmod m \quad (\text{A}\cdot 1)$$

$$r_k = \left\lfloor t/m^{n-(k+1)} \right\rfloor \bmod m \quad (\text{A}\cdot 2)$$
$$(k = 1, 2, \cdots, n-1)$$

Since $0 \le t < m^n$, we can express t as

$$\begin{aligned} t &= a_{n-1} + ma_{n-2} + ... + m^{n-2}a_1 + m^{n-1}a_0 \\ &= \sum_{i=0}^{n-1} m^{n-i-1}a_i \ \ (0 \le a_i < m). \end{aligned} \quad (\text{A}\cdot 3)$$

Substituting the above equation into Eq. (A·2) gives,

$$\begin{aligned} r_k &= \left\lfloor \sum_{i=0}^{n-1} m^{n-i-1}a_i/m^{n-(k+1)} \right\rfloor \bmod m \\ &= a_k \quad (1 \le k < n-1) \end{aligned} \quad (\text{A}\cdot 4)$$

and

$$t = \sum_{i=1}^{n-1} m^{n-i-1}r_i + m^{n-1}a_0. \quad (\text{A}\cdot 5)$$

Furthermore, substituting the above equation into Eq. (A·1) gives

$$\begin{aligned} r_0 &= \left(\sum_{i=0}^{n-1} \left\lfloor \left(\sum_{j=1}^{n-1} m^{n-j-1}r_j + m^{n-1}a_0\right)/m^i \right\rfloor\right) \bmod m \\ &= \left(\sum_{i=1}^{n-1} r_i + a_0\right) \bmod m. \end{aligned} \quad (\text{A}\cdot 6)$$

$0 \le a_0 < m, 0 \le r_0 < m$ leads

$$a_0 = \left(r_0 - \sum_{i=1}^{n-1} r_i\right) \bmod m. \quad (\text{A}\cdot 7)$$

Finally, we can obtain following equation for transmission order t.

$$t = \sum_{i=1}^{n-1} m^{n-i-1}r_i + m^{n-1}\left(\left(r_0 - \sum_{i=1}^{n-1} r_i\right) \bmod m\right). \quad (\text{A}\cdot 8)$$

We now investigate difference dr_k of transmission order t for code bits on a parity check line of direction r_k, which is specified by the coordinates $r_0, r_1, .., r_{k-1}, r_{k+1}, .., r_{n-1}$.
(A) In the case of $k \ne 0$,

$$\begin{aligned} dr_k &= t|_{r_k=i} - t|_{r_k=j} \quad (0 \le j < i \le m-1) \\ &= m^{n-k-1}(i-j) + \begin{cases} m^{n-1}(j-i) \\ m^{n-1}(j-i+m) \end{cases} \end{aligned} \quad (\text{A}\cdot 9)$$

where, the assumption of $i > j$ does not negate generality. dr_k has two values because of the cyclic structure of

t with period m^n. Since our purpose is to find the minimum difference, difference $|dr_k|$ must be the smaller of the two.

$$\min_{i,j} \left|m^{n-k-1}(i-j) + m^{n-1}(j-i)\right| = m^{n-1} - m^{n-k-1}, \quad (A\cdot 10)$$

which is given when $i - j = 1$, and

$$\min_{i,j} \left|m^{n-k-1}(i-j) + m^{n-1}(j-i+m)\right| = m^{n-1} + m^{n-k} - m^{n-k-1}, \quad (A\cdot 11)$$

given when $i - j = m - 1$. The former value is always the smaller of the two minima, so,

$$\min_{i,j} |dr_k| = m^{n-1} - m^{n-k-1} \ \ (k \geq 1). \quad (A\cdot 12)$$

(B) In the case of $k = 0$,

$$dr_0 = t|_{r_0=i} - t|_{r_0=j} \quad (0 \leq j < i \leq m-1) = \begin{cases} m^{n-1}(j-i) \\ m^{n-1}(j-i+m) \end{cases} \quad (A\cdot 13)$$

Similarly, the difference $|dr_0|$ must be the smaller of the two.

$$\min_{i,j} \left|m^{n-1}(j-i)\right| = m^{n-1}, \quad (A\cdot 14)$$

which is given when $i - j = 1$, and

$$\min_{i,j} \left|m^{n-1}(j-i+m)\right| = m^{n-1}, \quad (A\cdot 15)$$

given when $i - j = m - 1$. The above two minima are the same so

$$\min_{i,j} |dr_k| = m^{n-1} \ \ (k = 0). \quad (A\cdot 16)$$

In total, the minimum distance is given by

$$\min_{k=0,1,..,n-1} |dr_k| = m^{n-1} - m^{n-2}. \quad (A\cdot 17)$$

We note that the minimum difference of transmission order t on parity check lines varies according to the direction of parity line. This means that the definition of a parity check in a code block is not satisfied, so the transmission order could not be optimum.

Here we must remember that the number of parity check lines with a certain direction is m^{n-1} and that each line contains m digits out of m^n digits in the code. For these m^{n-1} parity lines, we can isolate the transmission order perfectly on each line by selecting t's every m^{n-1} digits. In that case, the minimum difference of t's is m^{n-1}. In other words, the minimum difference never exceeds m^{n-1}.

Since the number of such perfect selections is m^{n-1} and they are exhausted for above parity lines of the direction, the minimum difference can not reach m^{n-1} in parity lines with other directions.

In conclusion, we have not obtained a rigorous proof that confirms the transmission order yielded by Eq. (1) to be the optimum. Instead we consider it to be quasi-optimum.

Toshio Fukuta received B.E. and M.S. degrees from Nagoya Institute of Technology, Nagoya, Japan in 2001 and 2003, respectively. He is researching Error Correcting Codes in the Institute.

Yuuichi Hamasuna received B.E. (1991) and M.S. (1993) degrees in Electronic Engineering from Saga University, Saga, Japan. He joined DDS Inc. in 1995 and is conducting R&D activities on Error Correction Technologies.

Ichi Takumi received B.E. and M.S. degrees from Nagoya Institute of Technology, Nagoya, Japan in 1982 and 1984, respectively, both in electronics engineering. After graduation he joined Oki Electric Co. He has a Doctor of Eng. degree from Nagoya Institute of Technology. Since December 1985, he has been with the Nagoya Institute of Technology, where he is now a professor at the College of Shikumi. His current research interests include digital signal processing and digital communications.

Masayasu Hata graduated in 1958 from the Department of Electronic Engineering, Faculty of Engineering, Nagoya Institute of Technology, and is affiliated with Oki Electric Co. He has a Doctor of Eng. degree from Tokyo Institute of Technology. He was engaged in the R&D of digital communication systems, application of electronic circuits and millimeter wave communication equipment. He retired from Oki Electric Co. in 1985 and became a Professor of Nagoya Institute of Technology. Since 2002 he has been with Chubu University as a Professor. He is currently researching digital signal processing techniques and information communication.

Takahiro Nakanishi received B.E. and M.S. degrees from Nagoya Institute of Technology, Nagoya, Japan in 1993 and 1995, respectively, both in electronic and information engineering. He has a Doctor of Eng. degree from Nagoya Institute of Technology. Since 2003, he has been with Gunma University.