# Performance Investigation of Reduced Complexity Bit-Flipping using Variable Thresholds and Noise Perturbation

Julian Webber, *Member, IEEE,* Toshihiko Nishimura, *Member, IEEE,* Takeo Ohgane, *Member, IEEE,* and Yasutaka Ogawa, *Fellow, IEEE*

***Abstract*—The near Shannon capacity approaching low-density parity-check (LDPC) linear block codes are now in widespread use in modern systems including the long term evolution advanced (LTE-A) cellular, 802.11n Wi-Fi and DVB-S2 satellite communications standards. The decoders based on the iterative belief propagation algorithm provide near optimum performance but also have very high computational complexity. Therefore significant research has recently focused on reduced complexity architectures based on the group of so-called bit-flipping algorithms. In the basic bit-flipping algorithm the number of failed parity checks for each bit is computed and the bit with the maximum failed parity checks is inverted. Inverting bits above a certain threshold removes the complexity involved with a maximum-search and adaptive thresholds on each bit can further reduce the computation overhead. The criteria for threshold updates affects the error and convergence performances. Here, we describe a low-complexity architecture that has two (or more) decoder branches each with a different threshold scaling factor and select the threshold and bits at each iteration from the branch with the lowest syndrome sum. We then investigate the effect of adding a random Uniform or Gaussian noise perturbation to the threshold in order to reduce the average iteration count further in order to provide the opportunity to escape from stuck decoding states.**

***Index Terms*—bit-flip algorithm, gradient-descent, reduced-complexity, noise perturbation, LDPC decoding.**

## I. Introduction

Gallager first proposed LDPC codes in his 1963 PhD thesis [1]. However until the mid-1990s, when they were rediscovered by MacKay and Neal [2] and also coinciding with the availability of powerful low-cost digital signal processors, their use was largely forgotten. Near-optimum decoding performance can be achieved with the soft-decision belief propagation algorithm (BPA) that computes the marginal probabilities at the nodes on factor graphs [3]. The highest performance is achieved with long-length cycle-free codes and when the soft data at the decoder input is independent. However, the BPA also features a very high decoding complexity. At the other end of the complexity spectrum reside the bit-flipping algorithms (BFA) which are aimed at low-power portable applications. The average number of iterations in the BFA is higher than that of the BPA, but at each iteration a considerably smaller number of computations are required which can be computed in parallel, and the overall complexity is much less onerous.

In previous work we investigated the performance of a low complexity dual-scaling factor bit-flipping decoder with adaptive thresholds [4]. The addition of adding a simple noise perturbation to escape a stuck decoding state was proposed and initially investigated. In this paper we have extended the research to examine in more detail the effects of the noise perturbation on both the BER and iteration count. In particular, we investigate increasing the perturbation multiplication coefficient as well as the iteration delay size before which the perturbation is applied.

The paper is organized as follows: The belief propagation and bit-flipping algorithms are discussed in Section II. The adaptive bit-flipping technique together along with single and multiple-branch scaling factor decoders are introduced in Section III. The addition of a noise perturbation to escape a stuck-state is proposed and the effect of a noise coefficient multiplier and a sliding window counter concept are analysed in Section IV. Bit error and iteration count performance results are discussed in Section V and a conclusion is finally drawn in Section VI.

## II. Decoding algorithms

The received sample vector is $\mathbf{y} = (y_1, y_2, ..., y_N)$ and the hard decoded bits are given by $\mathbf{x} \in \{+1, -1\}$. A sparse parity check matrix $\mathbf{H}$ is of size $M \times N$, where the number of check nodes is $M$ and the number of bits is $N$ [5]. The non-zero elements in column $j$ are written as $M(j)$ and the non-zero elements in row $i$ of $\mathbf{H}$ are $N(i)$. The syndrome associated with the $m$-th check node is denoted as $s_m$ or equivalently the bipolar syndrome is written as $\prod_{j\in N(i)} x_j$. It is often instructive to graphically represent the parity check matrix by a bipartite Tanner graph [6] with $N$ variable nodes (number of bits in the codeword) and $M$ check nodes (number of parity bits) (Fig. 1).

### A. Belief Propagation

The belief propagation algorithm is based on the concept of exchanging probabilities via 'messages' between check nodes and variable nodes which generally become more accurate with each iteration. The message passed from the check node to the variable node is the probability that the variable node

J. Webber is with the Graduate School of Information Science and Technology, Hokkaido University, Kita 14, Nishi 9, Kita-ku, Sapporo 060-0814, Japan. e-mail: (julian.webber@m-icl.ist.hokudai.ac.jp).

T. Nishimura, T. Ohgane and Y. Ogawa are with the Graduate School of Information Science and Technology, Hokkaido University, Kita 14, Nishi 9, Kita-ku, Sapporo 060-0814, Japan. e-mail: (nishimura@ist.hokudai.ac.jp).

Manuscript received June 27, 2012; revised January 1, 2000.

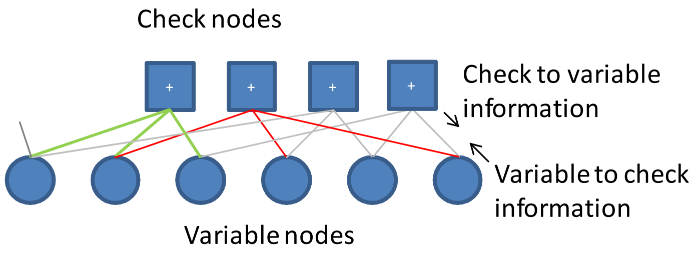

Fig. 1. LDPC decoding using Tanner graph representation.

has a certain value given all the messages passed to that check node in the preceding iteration from message nodes except the variable node itself. At the same time, the message passed from a variable node to a check node is the probabilty that it has a certain value given its observed value, and all values passed to it in the previous iteration from check nodes connected to it other than the check node itself [7]. Usually the probabilities are represented by log-likelihood ratios. The algorithm provides optimum performance but at the cost of very high computational complexity.

## B. Bit-flipping algorithms

The sign of a single bit $n$ with a maximum inversion function, $E_n$ is inverted at each iteration in the basic BFA. A weighted BF (WBF) algorithm was proposed by Kou et al. [8] that incorporates a term related to the energy of the least reliable bit involved in each check. This however necessitates a search over the complete slot-length which contributes to an increased complexity and delay. The WBF is expressed as

$$E_n = \sum_{i\in M(n)} \{\min_{j\in N(i)}|y_j|\} \prod_{j\in N(i)} x_j, \tag{1}$$

where $y_m^{min}$ is the least reliable message node associated with the $m$-th check.

Given that two message bits have contributed to a failed parity check, it is the message with the highest energy $|y_j|$ that is deemed most likely to be correct. In the reliability ratio algorithm [9] a normalization factor $R_{ij} = \beta\frac{|y_j|}{|y_i^{max}|}$ is calculated where $|y_i^{max}|$ is the highest soft magnitude of all message nodes associated with the $m$-th check (Eq. 2). The normalization improves the performance with respect to the WBF algorithm but the division operation in the initialization stage contributes to an increased complexity. An implementation-efficient reliability ratio WBF (IRRWBF) algorithm was subsequently proposed which lowered the complexity particularly when the iteration count was low and the code length was high [10].

$$E_n = \sum_{i\in M(n)} \prod_{j\in N(i)} /R_{ij}. \tag{2}$$

where, $R_{ij} = \beta\frac{|y_j|}{|y_i^{max}|}$. In addition to the syndrome sum the gradient descent bit flipping (GDBF) algorithm [11] includes a correlation term between the initial received soft and hard decision which should be large and positive for a correctly

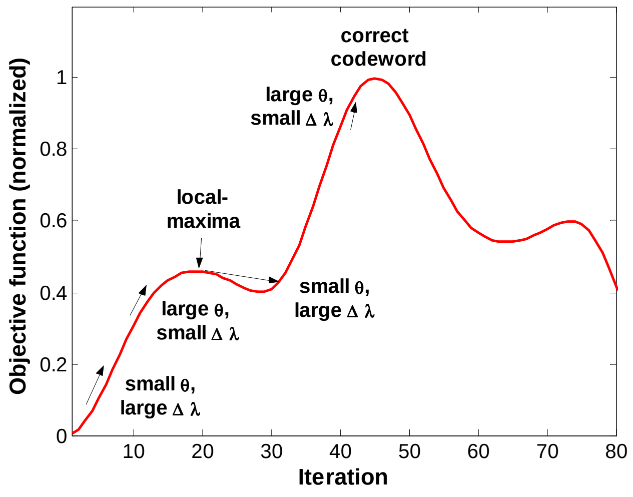

Fig. 2. Concept of the stuck decoding states. The step-size is reduced on reaching a local-maxima and increased on escaping it. Eventually the correct codeword is found.

estimated bit [11].

$$\Delta_n^{GD}(\mathbf{x}) = \sum_{i\in M(n)} \prod_{j\in N(i)} x_j + x_n y_n. \tag{3}$$

A bit is inverted if $\Delta_n^{GD}(\mathbf{x})$ is less than a global threshold. An objective function indicates the state of the algorithm at each iteration and is defined as

$$f^{GD}(\mathbf{x}) = \sum_{i\in M(n)} \prod_{j\in N(i)} x_j + \sum_{n=1}^{N} x_n y_n. \tag{4}$$

By monitoring the objective function it was proposed to switch from a large step-size (multiple bit-flips) to a small step-size if a stuck decoding state was detected. On such occasions a single small-descent of the objective function using a lower set threshold with a random component was shown experimentally to improve the performance. On escaping a local-maxima the step-size can then increase and eventually the correct codeword is found (Fig. 2).

## III. Adaptive flipping thresholds

Recently adaptive thresholds for codes with a large number of checks per bit were investigated by Cho [12]. The benefit gained from inverting erroneous bits minus the loss from inverting correct bits was defined as the ‘flipping-gain’. A known monotonically increasing number of errors was injected into a packet, and the threshold range $p$ that produced a flipping-gain greater than one was calculated. It was observed that there was indeed an optimal threshold that resulted in a minimum number of iterations. It was conjectured that a similar analysis could be applied to parity check codes containing a smaller number of checks per bit, (e.g. as $M$=3), using non-integer thresholds (e.g. 1.33) and using a probabilistic strategy (e.g. thresholds 1.0, 1.0, 2.0).

In [13], an adaptive threshold $\lambda_n$ was set on each bit. The sign of a bit was inverted if its inversion function was below

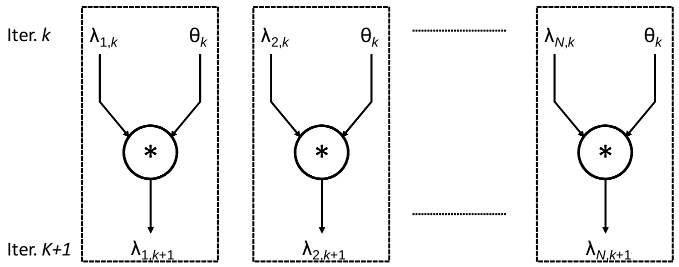

Fig. 3. The updated threshold $\lambda_{n,k+1}$ for bit $n$ at iteration $k+1$ is computed by summing one or more shifted versions of the previous threshold $\lambda_{n,k}$.

TABLE I
THRESHOLD SCALING VALUES, $\theta$ AND PARTIAL COMPONENTS.

| $\theta$ (decimal) | $d_X$ | $d_Y$ | $\theta_A$ | $\theta_B$ |
|---|---|---|---|---|
| 0.5 | 1 | 0 | 1/2 | 0 |
| 0.625 | 1 | 3 | 1/2 | +1/8 |
| 0.75 | 1 | 2 | 1/2 | +1/4 |
| 0.875 | 0 | 3 | 1 | -1/8 |
| 0.9375 | 0 | 4 | 1 | -1/16 |
| 0.969 | 0 | 5 | 1 | -1/32 |

the particular threshold (5), and otherwise it's threshold was lowered.

$$E_n(k) \leq \lambda_n(k). \tag{5}$$

The threshold update rule is written as (6)

$$\lambda_n(k+1) = \theta\ \lambda_n(k), \tag{6}$$

where $k$ is the iteration number and $\theta$ is a threshold scaling value. With this technique the algorithm moves naturally from many to few bit-flips as the threshold is successively lowered. The computational complexity is lowered as each bit-processing operation (i.e. scaling / inversion) is computed independently of other bits. Furthermore, the physical hardware multiplier can be replaced a simple shift if the scale factor is chosen to be proportional to a power of two (Fig. 3).

## A. Threshold scaling factor

The scaling factor $\theta$ in this paper is expressed as the sum of two fractional parts as :

$$\theta = 2^{-d_X} + 2^{-d_Y}, \tag{7}$$

where $d_X$ and $d_Y$ are integers. The range of values for $\theta$ that were investigated in this work along with the required shifts are listed in TABLE I. To generate the threshold update $\lambda_n(k+1) = 0.625\lambda_n(k)$ for example, the current threshold $\lambda_n(k)$ is shifted once 1 place (i.e. 0.5), once 3 places (i.e. 0.125), and the two partial products are summed. A small value of $\theta$ (e.g. 0.5) clearly equates to a large step-size and a large value of $\theta$ (e.g. 0.969) corresponds to a small step-size. By applying more shift addition operations any arbitrary scaling factor can be created, though for simplicity two additions were demonstrated in this work.

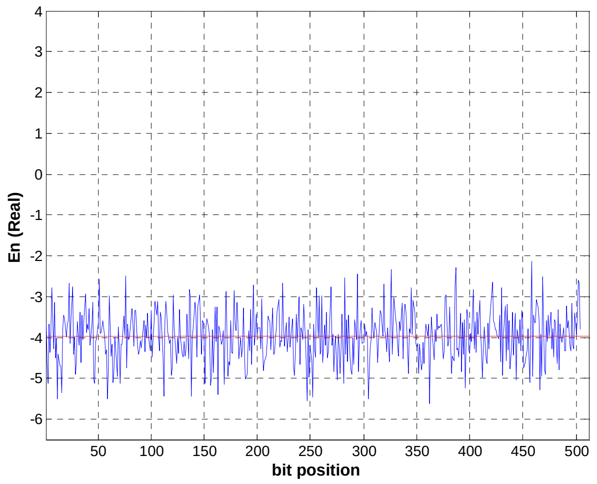

Fig. 4. Snapshot of the inversion function $E_n$ of each bit at iteration 60 ($\theta$=0.97). The mean value is indicated by the red horizontal line.

TABLE II
LDPC CODE PROPERTIES.

| Code | Name [15] | Rows | Cols. | Col. deg. |
|---|---|---|---|---|
| 1 | PEGReg504x1008 | 504 | 1008 | 3 |
| 2 | PEGReg252x504 | 252 | 504 | 3 |

## B. Single scaling-factor

In this paper two progressive edge growth (PEG) regular LDPC codes are investigated with their properties listed in Table II. PEG codes are created with the aim to maximize the girth (i.e. length of the shortest cycle) and therefore generally provide good performance [14]. The inversion function $E_n$ recorded from a random trial at iteration 60 is plotted in Fig. 4. The bits whose inversion function $E_n$ are above the threshold are flipped at each iteration. On flipping the value of the bits' inversion function will change and become more negative as its' probability of being correct increases.

## C. Multiple scaling-factors / branches

In this work a novel dual-scaling switching algorithm (DSA) is proposed that jointly uses a fine and a coarse step-size decoder. The estimated hard bits from the decode-branch that have the minimum syndrome sum are passed to both decoder branches for the start of the next iteration (Fig. 5). The optional noise perturbation block is discussed further in the next section. The algorithm is sumarized as

**STAGE 1-** Initialize the inversion thresholds $\lambda_n$=$\lambda_0$, $\forall\ n$. Set State=1.
**STAGE 2-** Compute the inversion function $E_n$, for $m = 1, 2, ..., M$.
**STAGE 3-** If $E_n(\theta_2) \geq \lambda_n$, Flip bit $n$. Else $\lambda_n$=$\theta_2\lambda_{n-1}$. Optionally add noise perturbation to $\lambda_n$.
**STAGE 4-** If state=1, Repeat STAGE 3 for $\theta$=$\theta_1$.

**STAGE 5-** If $\Sigma s_m(\theta_1) \geq \Sigma s_m(\theta_2)$, Set state=2; $\lambda_n$=$\lambda_n(\theta_2)$, $\forall$ $n$ and **x**=**x**($\theta_2$); Deactivate branch no.1 to reduce energy consumption. Else $\lambda_n$=$\lambda_n(\theta_1)$, $\forall$ $n$ and **x**=**x**($\theta_1$).

**STAGE 6-** Terminate if $\Sigma s_m = 0$ or maximum iterations reached.

Else Goto STAGE 2.

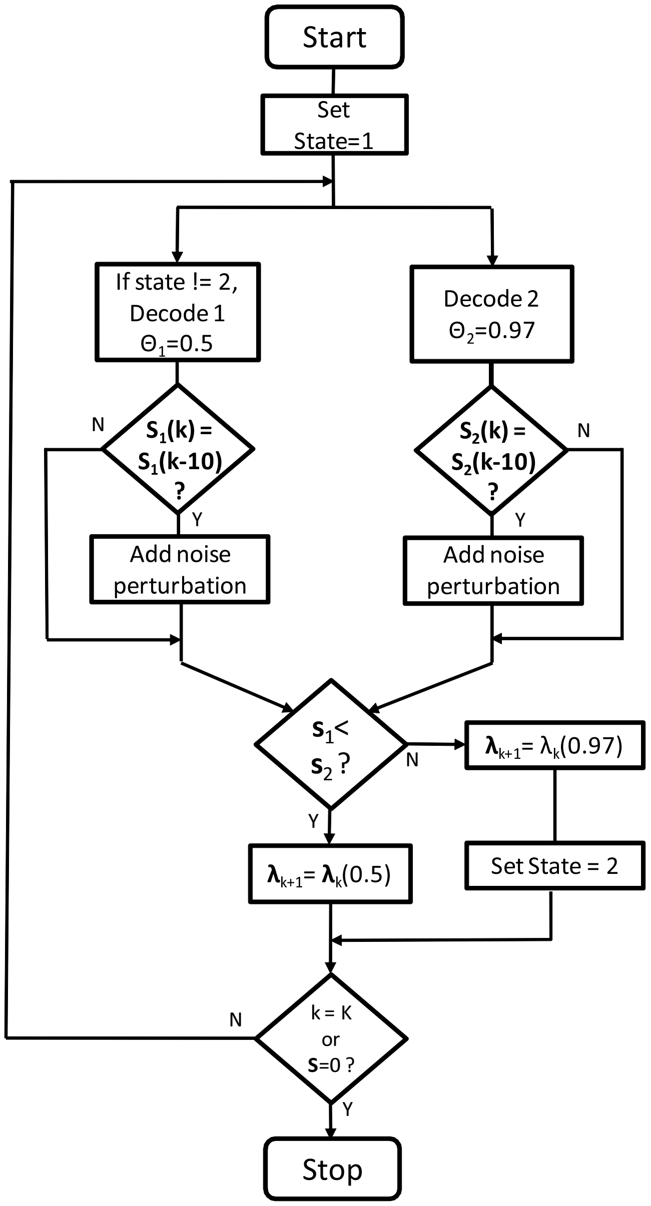

Fig. 5. Proposed dual scaling threshold algorithm flow chart showing optional noise perturbation unit.

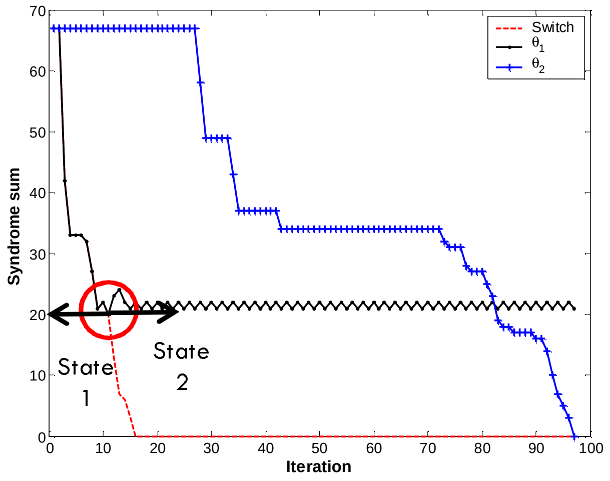

Fig. 6. Graph showing the syndrome sum versus iteration count. Single scaling factor $\theta_1$=0.5 and $\theta_2$=0.97. 'Switch' indicates the dual branch decoder.

The number of failed parity checks versus iteration number for the DSA is shown in Fig. 6. The solid line (marked '.') plots the syndrome sum when $\theta_1$=0.5 and the solid line (marked '+') when $\theta_2$=0.969. The number of failed checks decreased to zero by iteration 98 using the small step-size. However, by switching at iteration 11 from $\theta_1$=0.5 to $\theta_2$=0.969 when the syndrome sum was lower on that branch, the number of failed checks was rapidly reduced to zero by the 18th iteration.

Although the overall complexity is increased relative to a single branch decoder, this can be limited by deactivating the $\theta_1$ branch once the $\theta_2$ branch produces less errors. The optimum scaling factor is a function of the modulation, SNR, and particular code characteristic which makes calculating an optimum value very complex and an off-line statistical approach was made in [13] through experiment trials.

A further benefit of the dual-scaling detector proposed here is that both a real-time or off-line threshold calculation are avoided. A three-branch (three-scaling factors) detector was also evaluated but achieved only a small reduction in the iteration count with respect to the dual-scaling decoder and hence, due to the additional complexity, was not considered further in this work.

To reduced complexity an early-stopping algorithm was investigated in which processing was stopped if $f(k)$=$f(k-10)$ or $f(k)$=$f(k-11)$. It was necessary to apply the two conditions in case $f(k)$ oscillates between two values such as seen by the black zig-zag line in Fig. 6.

## IV. Noise perturbation

The local minima are a feature of the gradient descent algorithm where trapped search points occur but the estimated sequence does not correspond to the transmitted code-word. In such a situation we investigate the technique of inserting a small random perturbation to the threshold of each bit whenever the syndrome sum does not change over a sliding window of the previous $W$ iterations. When this condition is satisfied a new random noise sample $z_n$ is added to the current value of $\lambda_n$ as shown by

$$\lambda_{n+1} = \lambda_n + K z_n, \tag{8}$$

where $z_n$ is a random Uniform or Gaussian sample for bit $n$ and $K$ is a noise perturbation coefficient typically between 1.2 and 2.4 for Uniform noise and 0.7 to 1.3 for Gaussian noise.

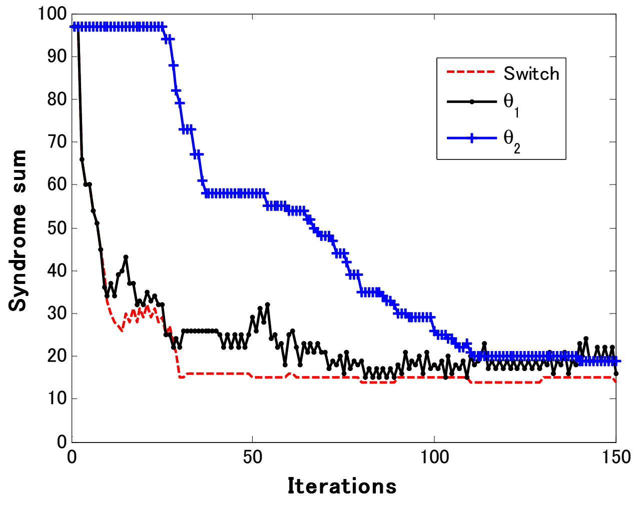

Fig. 7. Graph showing the syndrome sum against iteration with noise perturbation at iterations: 50, 60, ... 150.

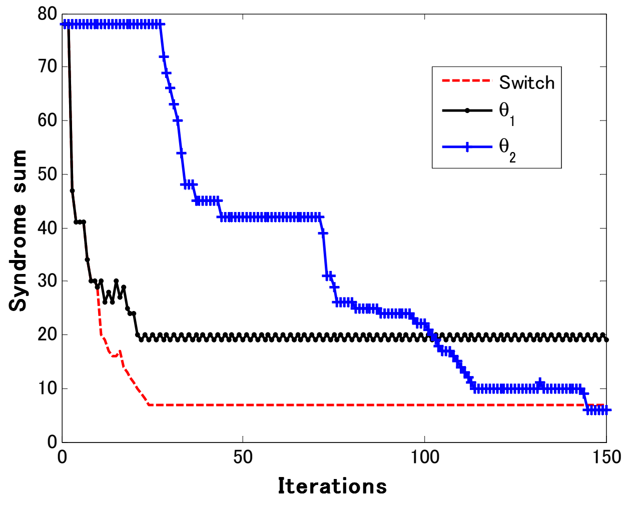

Fig. 8. Plot showning syndrome sum without added random perturbation.

## A. Temporal study

We first investigate through experiment the instantaneous effect on the syndrome sum of adding a noise perturbation. In Fig. 7 Uniform noise was injected at iteration 50 and subsequently at intervals of 10 iterations thereafter. We observe a small but noticeable lowering in the number of syndrome errors at iterations 50, 80 and 110. Due to the stochastic nature of the noise perturbation, there will be occasions when there is no improvement such as at iteration 100 for example.

In the absence of a random perturbation, the syndrome sum recorded during one decoding trial, is plotted in Fig. 8. Next, using the same received signal sequence $\mathbf{y}$ for fairness, the syndrome sum is computed in a trial with perturbation permitted (Fig. 9). The benefit of the the random perturbation can clearly be seen where a stuck decoding state was successfully exited and the syndrome sum reduced to zero by iteration 66. On adding the perturbation, the particular bits whose current inversion function is nearest to the threshold are most likely to be inverted and the noise coefficient $K$ controls the number of additional bits that have the potential to be inverted. Note that a program 'break' was used within the iteration loop and in parallel the three independent decoders were ran generating the red, blue, black curves. Once the syndrome sum was zero on the red dual-threshold detector, the loop was exited and so the trajectories for the blue and black curves are not plotted beyond iteration 66, even though they would continue in normal operation.

In the following two subsections we study in more detail the effects of the noise coefficient multiplier $K$ and the sliding window counter size $W$. In particular, we aim to find optimum values through experimentation across a range of practical signal to noise ratios.

## B. Noise coefficient

A very low noise coefficient $K$ would result in applying no perturbation and too high a value would cause the flipping threshold to be adjusted too quickly with a similar effect to that of a large step-size $\theta$. Thus an optimum coefficient range would be expected and an experiment was set-up to find the range. The number of iterations versus the noise coefficient were recorded across the SNR range from 2-4 dB. It is shown in Fig.10 that in the transition region the optimum value is about 1.6 for both Code 1 (top) and Code 2 (bottom).

A comparison between Gaussian and Uniform noise perturbations is shown in Fig.11. It was found that in terms of the number of iterations, the same performance can be achieved when $K$ was set to 1.0 when applying a Gaussian source and 1.5 for Uniform sources. This result was valid for both LDPC codes Code 1 and Code 2.

## C. Sliding window counter size

The noise perturbation is only required when a stuck decoding state is reached. This can be detected when there is

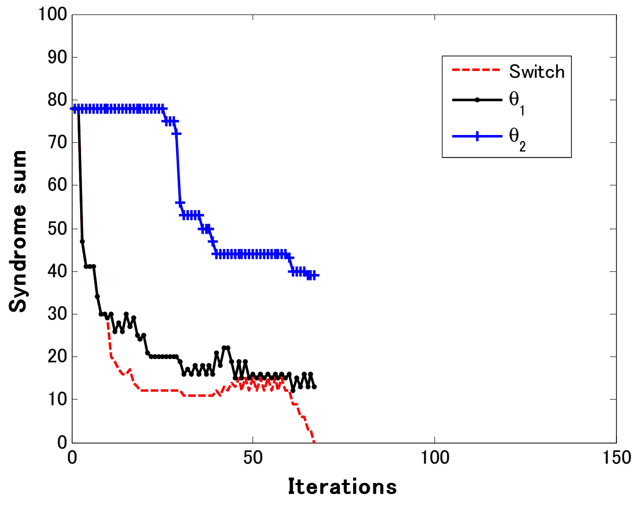

Fig. 9. Plot showing syndrome sum with added perturbation. This trial used the exact same received soft data, $\mathbf{y}$ as in Fig. 8 for fairness.

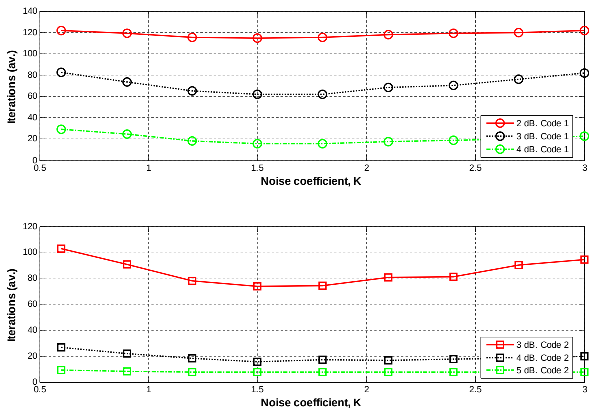
Fig. 10. Effect of the noise coefficient $K$ on iteration count for Code 1 (top) and Code 2 (bottom).

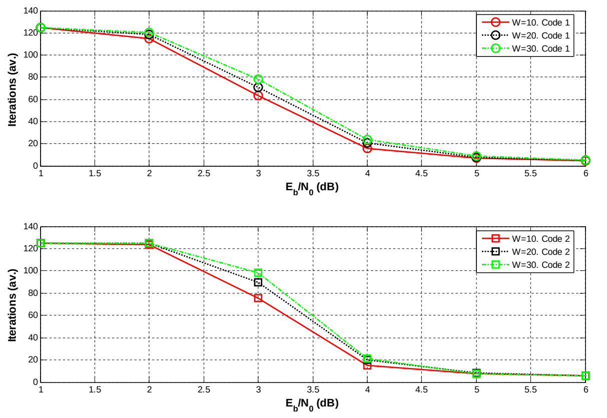
Fig. 12. Effect of sliding window size $W$ on iteration count for Code 1 (top) and Code 2 (bottom).

no improvement in the syndrome sum despite the threshold being lowered over a number of iterations. In such a situation a counter is incremented at each iteration and otherwise reset to zero if the threshold was successfully lowered. If the counter reaches a set number, referred to as the sliding window counter size $W$ then a noise perturbation is added.

If a perturbation is added too soon the benefit of a small step-size is not realized as there is little opportunity to slowly lower the threshold. On the other hand, if the perturbation is added too late then the algorithm takes an unnecessary long time and wasteful iterations are required increasing the power consumption. It was therefore thought that an optimum range for the sliding window counter would exist.

In this experiment the counter is varied between 10 and 30 iterations and the maximum number of iterations was set at 125. The resulting number of iterations until convergence (or algorithm termination) is shown in Fig.12 for Code 1 (top) and for Code 2 (bottom). The effect of the sliding window length $W$ on the average number of iterations as the SNR increases is shown in Fig.13 Code 1 (top) and Code 2 (bottom).

The effect of the noise coefficient $K$ on the average number of iterations for various SNR is shown for Code 1 in Fig.14 for (top) Gaussian and (bottom) Uniform perturbation noise sources. The optimum value is observed when the BER gradient is at its largest value. For the case of Code 1 we can clearly see a minimum number of iterations exist at between 2 and 4 dB SNR for both Gaussian and Uniform noise sources. In low SNR or as the algorithm approaches convergence there is clearly no benefit in adding the perturbation. Further work could investigate the optimum value for each threshold factors.

## V. Error performances

The error performances of the various bit flipping algorithms were evaluated in an AWGN channel through simulation. The maximum number of iterations was set at 150 with BPSK modulation. In the BER evaluation the sliding window counter size, $W$ was set at 10 iterations. Although the IRRWBF algorithm performed the best it also has the highest complexity. The BER depends on the particular value of $\theta$ when the

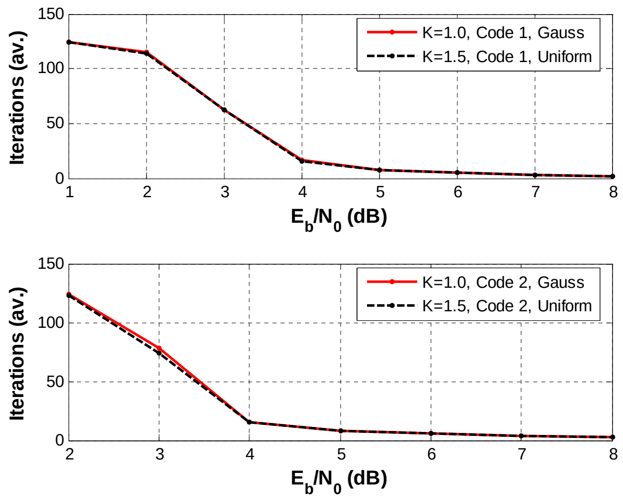
Fig. 11. Comparison of Gaussian and Uniform noise perturbations for (top) Code 1 and (bottom) Code 2.

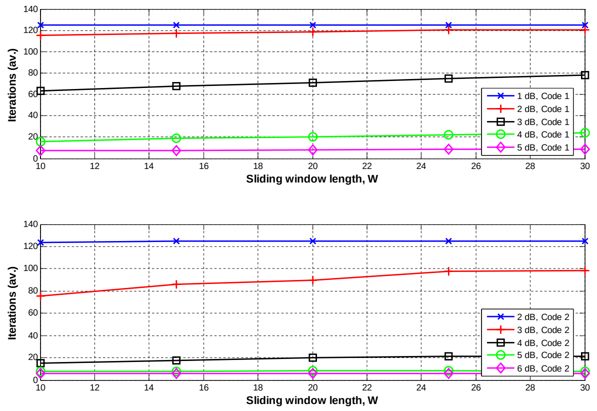
Fig. 13. Effect of sliding window length on average number of iterations (top) Code 1 and (bottom) Code 2.

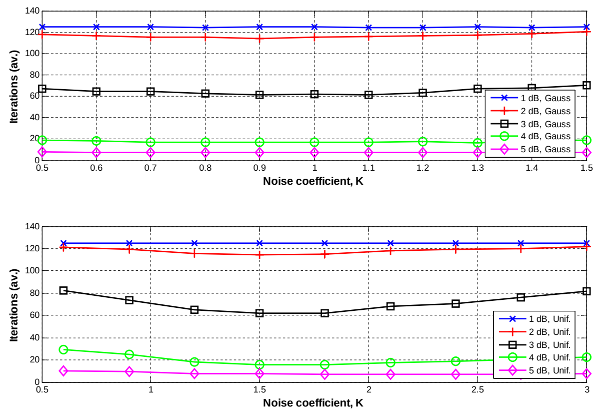

Fig. 14. Effect of Noise coefficient $K$ on average number of iterations for various SNR (top) Gaussian and (bottom) Uniform noise using Code 1.

number of iterations is limited as shown in Fig. 15. When $\theta$=0.5, more bits are inverted on average per iteration and the overall performance is worse than when $\theta$=0.97. The proposed DSA with $\theta$=0.625/0.97 performed as well as well as the one with a single $\theta$=0.97 but its advantage is the reduced iteration count as seen in Fig. 16 where the average number required is lowered by 41% from 66 to 39 at 4 dB SNR. The number of iterations of the 'Dual early-stop' algorithm was similarly reduced to 17 at an $E_b/N_0$ of 4 dB.

The performance of the random perturbation decoders are denoted by '+ perturb.'. At the BER of $10^{-3}$, an improvement of 0.5 dB was obtained for the dual scaling decoder with perturbation (Fig. 17). As the fine step-size decoder slowly converges to the desired solution there is only a small improvement in BER. However, at 3 dB $E_b/N_0$ the number of iterations was reduced on average by (16.4, 9.2, 2.0) for the (dual-scale, $\theta$=0.63 and $\theta$=0.97) decoders respectively (Fig.18). The corresponding savings were (11.0, 10.1, 2.6) iterations at4 dB $E_b/N_0$.

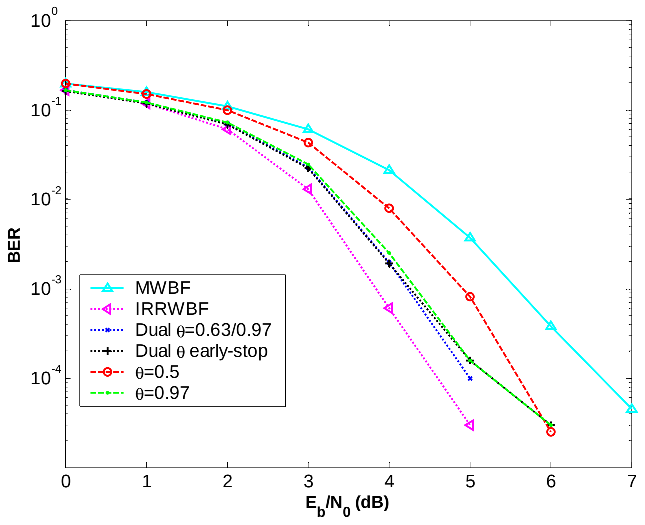

Fig. 15. BER versus $E_b/N_0$ (dB) with LDPC Code 1 in AWGN.

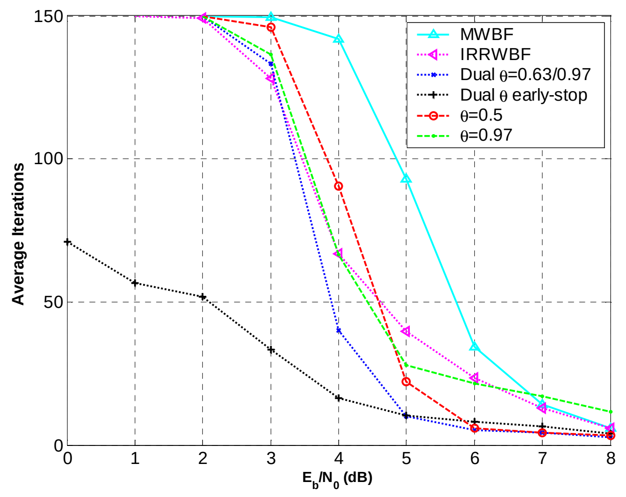

Fig. 16. Average number of iteration versus $E_b/N_0$ (dB) for Code 1.

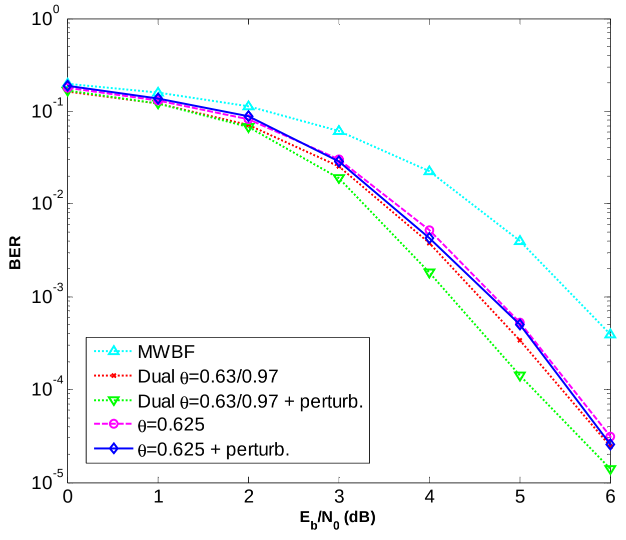

Fig. 17. BER versus $E_b/N_0$ (dB) for LDPC Code 2 with noise perturbation in AWGN.

## VI. Conclusion

The error and convergence rates of adaptive threshold bit-flipping algorithms have been studied. A dual-scaling architecture was proposed that applies the hard bit estimates and thresholds at each iteration from either the fine or coarse step-size decode branch depending on its resepective syndrome sum. The dual-scale architecture with $\theta$=0.63/0.97 performed as well as that of the single step-size $\theta$=0.97 decoder but the average number of iterations was reduced by 41% at 4 dB $E_b/N_0$. By adding a small noise perturbation the average iteration count of the dual-scale detector could further be reduced by 11 cycles at 4 dB $E_b/N_0$. An optimum value for the perturbation coefficient was found to be 1.6 for Uniform noise and applied after a delay of 20 iterations. Further work could investigate the optimum value as a function of the threshold step-size.

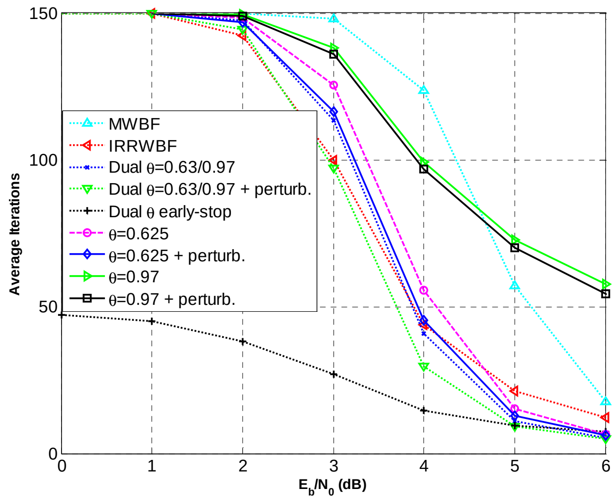

Fig. 18. Average number of iteration versus $E_b/N_0$ (dB) for Code 2 with perturbation.

## Acknowledgment

The authors gratefully acknowledge the support of the GCOE program, Graduate School of Information Science and Technology, Hokkaido University.

## References

[1] R. Gallager, "Low Density Parity Check Codes (LDPC)," *Mass. Inst. Tech. Press*, Phd. Thesis, 1963.

[2] D. MacKay and R. Neal, 'Good error-correcting codes based on very sparse matrices," *Cryptography and Coding 5th IMA conference*, Jul. 1995.

[3] W. Chung, and J. Cruz, "An improved belief-propagation decoder for LDPC-coded partial-response channels," *IEEE Trans. Magnetics*, vol. 46, no. 7, pp. 2639-2648, Jul. 2010.

[4] J. Webber, T. Nishimura, T. Ohgane, and Y. Ogawa, "A study on adaptive thresholds for reduced complexity bit-flip decoding," *International Conference on Advanced Communications Technology (ICACT'12)*, Phoenix Park, Pyeongchang, Korea, Feb. 2012.

[5] D. J. MacKay, "Information thoery, inference, and learning algorithms," *Cambridge University Press*, 2003.

[6] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533-547, Sep. 1981.

[7] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Analysis of low density codes and improved designs using regular graphs," *IEEE Trans. on Inform. Theory*, vol. 47, pp. 585-598, 2001.

[8] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, no. 7, pp. 2711-2736, Nov. 2001.

[9] F. Guo and L. Hanzo, "Reliability ratio based weighted bit-flipping decoding for LDPC codes," *IEE Electron. Lett.*, vol. 40, no. 21, pp. 1356-1358, Oct. 2004.

[10] C. H. Lee and W. Wolf "Implementation-efficient reliability ratio based weighted bit-flipping decoding for LDPC codes," *IEE Electronics Lett.* vol. 41, no. 13, pp. 755-757, Jun. 2005.

[11] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit-flipping algorithms for decoding LDPC codes," *International Symposium on Information Theory and its Applications (ISITA'08).*, Auckland, New Zealand, Dec. 2008.

[12] J. Cho and W. Sung, "Adaptive threshold technique for bit-flipping decoding of low-density parity-check codes," *IEEE Comm. Letts.*, pp. 857-859, col. 14, no. 9, Sep. 2010.

[13] M. Ismail, I. Ahmed, J. Coon, S. Armour, T. Kocak, and J.P. McGeehan, "Low latency low power bit flipping algorithms for LDPC decoding," *IEEE PIMRC10*, Sep. 2010.

[14] E. Psota, and L. Perez, "Iterative construction of regular LDPC codes from independent tree-based minimum distance bounds," *IEEE Comm. Letts.*, vol. 15, no. 3, Mar. 2011.

[15] D. MacKay, "Encyclopedia of sparse graph codes [Online]," *URL: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html*, Accessed Feb. 2011.

**Julian WEBBER** Julian WEBBER received the M.Eng. and Ph.D. degrees from the University of Bristol, UK in 1996 and 2004 respectively. From 1996 to 1998, he was with Texas Instruments working on ASIC and DSP. From 2001-07 he was a Research Fellow at Bristol University working on real-time MIMO-OFDM test beds and digital pre-distortion systems. Since 2007, he has been with Hokkaido University, where he is a Research Fellow. His current research interests are in LDPC and MIMO signal processing. Dr Webber is a member of the IEEE.

**Toshihiko NISHIMURA** Toshihiko NISHIMURA received the B.S. and M.S. degrees in physics and Ph.D. degree in electronics engineering from Hokkaido University, Sapporo, Japan, in 1992, 1994, and 1997, respectively. In 1998, he joined the Graduate School of Engineering (reorganized to Graduate School of Information Science and Technology at present) at Hokkaido University, where he is currently an Assistant Professor of the Graduate School of Information Science and Technology. His current research interests are in MIMO systems using smart antenna techniques. Dr. Nishimura received the Young Researchers' Award of IEICE Japan in 2000, and the Best Paper Award from IEICE Japan in 2007. Dr. Nishimura is a member of the IEEE.

**Takeo OHGANE** Takeo OHGANE received the B.E., M.E., and Ph.D. degrees in electronics engineering from Hokkaido University, Sapporo, Japan, in 1984, 1986, and 1994, respectively. From 1986 to 1992, he was with Communications Research Laboratory, Ministry of Posts and Telecommunications. From 1992 to 1995, he was on assignment at ATR Optical and Radio Communications Research Laboratory. Since 1995, he has been with Hokkaido University, where he is an Associate Professor. During 2005-2006, he was at Centre for Communications Research, University of Bristol, U.K., as a Visiting Fellow. His interests are in MIMO signal processing for wireless communications. Dr. Ohgane received the IEEE AP-S Tokyo Chapter Young Engineer Award in 1993, the Young Researchers' Award of IEICE Japan in 1990, and the Best Paper Award from IEICE Japan in 2007. Dr. Ohgane is a member of the IEEE.

**Yasutaka OGAWA** Yasutaka OGAWA received the B.E., M.E. and Ph.D. degrees from Hokkaido University, Sapporo, Japan, in 1973, 1975, and 1978, respectively. Since 1979, he has been with Hokkaido University, where he is currently a Professor of the Graduate School of Information Science and Technology. During 1992-1993, he was with ElectroScience Laboratory, the Ohio State University, U.S.A., as a Visiting Scholar, on leave from Hokkaido University. His interests are in adaptive antennas, mobile communications, super-resolution techniques, and MIMO systems. Dr. Ogawa received the Young Researchers' Award of IEICE Japan in 1982, and the Best Paper Award from IEICE Japan in 2007. Dr. Ogawa is a Fellow of the IEEE.