PERFORMANCE ANALYSIS OF THE ADAPTIVE PARITY CHECK MATRIX BASED SOFT-DECISION DECODING ALGORITHM

Arshad Ahmed, Ralf Koetter, and Naresh R. Shanbhag

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308, West Main Street, Urbana, IL 61801.
Email: [ahmed4,koetter,shanbhag]@uiuc.edu

ABSTRACT

A gradient descent, iterative soft-decision algorithm for decoding Reed-Solomon codes using adaptive parity check matrices has been proposed recently. This algorithm outperforms all known Reed-Solomon soft-decoding algorithms at moderate SNR. However, many applications operate at a high SNR with frame error rate requirements in the range of $10^{-10} \sim 10^{-20}$. At these frame error rates, simulation based performance validation is prohibitive. In this paper, we present a model to analytically compute the soft-decoding algorithm performance. Using the insight obtained from this model, we propose a *low complexity*, *non-iterative* algorithm using adaptive parity check matrices, with a similar decoding performance as the original iterative algorithm. We also propose an extension to the non-iterative algorithm which improves on the decoding performance of the iterative algorithm.

1. INTRODUCTION

Reed-Solomon (RS) codes are a class of linear block codes that are *maximum distance separable* (MDS), *i.e*, they achieve the largest possible *minimum distance* between codewords for the allowed redundancy. In addition, for these codes, errors upto half the minimum distance can be efficiently corrected, using algorithms such as the Berlekamp-Massey algorithm [1] and the Euclid algorithm [1]. Consequently, RS codes are widely used for error-correction in communication systems. There has been substantial research directed toward developing algorithms which improve performance by decoding beyond half the minimum distance of the code. Most of these algorithms are *soft-decision* decoding algorithms in that they use the reliability of the received information in the decoding process. Well-known soft-decoding algorithms include generalized minimum distance decoding [2], Chase decoding [3], ordered statistics decoding [4], and algebraic soft-decision decoding [5]. The performance improvement of these algorithms over traditional decoding is obtained at the cost of increased decoding complexity.

This work was funded by NSF grant: CCR 00-73490

The adaptive parity check based soft-decoding (APC-SD) algorithm proposed in [6], is a gradient descent algorithm, based on a modified form of belief propagation [7] on the parity check matrix of the RS code. This paper deals with an expectation based analysis of the performance of this algorithm. This analysis motivates non-iterative variants, which either meet or exceed the performance of the original algorithm.

The rest of the paper is organized as follows. Section 2 describes the gradient descent based iterative soft-decoding algorithm proposed in [6]. Section 3 presents the analytical model for the performance of the APC-SD algorithm. Section 4 presents non-iterative algorithms based on the concept of adaptive parity check matrices. Finally, conclusions are given in Section 5.

2. ADAPTIVE PARITY CHECK BASED SOFT-DECISION DECODING

Let n denote the number of code-symbols and k denote the number of data symbols in a RS code, defined over a Galois field with 2^q elements. For this $[n, k]$ RS code over $GF(2^q)$, the parity check matrix $\mathbf{H}$, can be represented as:

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{(n-1)} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \alpha^{(n-k)} & \alpha^{2(n-k)} & \cdots & \alpha^{(n-k)(n-1)} \end{bmatrix} \quad (1)$$

Here α denotes the primitive element of $GF(2^q)$ [1]. Since each symbol in $GF(2^q)$ comprises q bits, the parity check matrix in (1) can be expressed as a binary matrix of size $(n-k)q \times nq$. An example of the binary parity check matrix for a $[7, 5]$ RS code over $GF(2^3)$ is given in (2).

Let the binary form of the RS codeword be represented as: $\mathbf{c} = [c_1, c_2, \ldots, c_{nq}]$. Assuming BPSK modulation and a AWGN channel, the received vector can be represented as $\mathbf{r} = [r_1, r_2, \ldots, r_{nq}]$, where $\mathbf{r} = 1 - 2\mathbf{c} + \eta$. Let $\mathbf{y} = [y_1, y_2, \ldots, y_{nq}]$, a real-valued random vector, denote the

0) Initialization:

Set step-size: μ, maximum iterations: N, iteration index: $j = 0$, and initial LLR vector $\mathbf{L}^{(0)}(\mathbf{y}) = \left(2/\sigma^2\right)\mathbf{r}$.

1) Parity Matrix Adaptation:

a) Order the LLR values with increasing reliability using absolute LLR values $|\mathbf{L}^{(j)}(\mathbf{y})|$.

b) Reduce to unit weight, $(n-k)q$ independent columns of $\mathbf{H}$ corresponding to $(n-k)q$ low-reliability bits; thus obtain $\mathbf{H}^{(j)}$.

2) Gradient Descent Update:

a) Generate the extrinsic information vector $\mathbf{L}_{ext}(\mathbf{y})$, using $\mathbf{H}^{(j)}$ and $\mathbf{L}^{(j)}(\mathbf{y})$.

b) Update the LLR vector as: $\mathbf{L}^{(j+1)}(\mathbf{y}) = \mathbf{L}^{(j)}(\mathbf{y}) + \mu\mathbf{L}_{ext}(\mathbf{y})$

3) Hard Decision:

$\hat{y_i} = \begin{cases} 0 & L^{(j+1)}(y_i) > 0 \\ 1 & L^{(j+1)}(y_i) \leq 0 \end{cases}$

4) Termination Branch:

If $j < N$ and $\mathbf{H} \cdot \hat{\mathbf{y}} \neq \mathbf{0}$ then increment j and return to Step 1; else output hard-decision bit vector: $\hat{\mathbf{y}}$.

Fig. 1. Pseudo-code of the Adaptive Parity Check Soft-Decoding (APC-SD) algorithm.

soft-estimate of the transmitted codeword. The initial log-likelihood ratio (LLR) vector associated with this received vector is given by $\mathbf{L}(\mathbf{y}) = \left(2/\sigma^2\right)\mathbf{r}$.

$$\mathbf{H}_{[7,5]} = \begin{bmatrix} 100 & 010 & 101 & 110 & 111 & 011 & 001 \\ 010 & 101 & 110 & 111 & 011 & 001 & 100 \\ 001 & 100 & 010 & 101 & 110 & 111 & 011 \\ 100 & 101 & 111 & 001 & 010 & 110 & 011 \\ 010 & 110 & 011 & 100 & 101 & 111 & 001 \\ 001 & 010 & 110 & 011 & 100 & 101 & 111 \end{bmatrix} \quad (2)$$

The binary parity check matrix $\mathbf{H}$ and the initial LLR vector $\mathbf{L}(\mathbf{y})$ are the two inputs required to run a message passing algorithm [7], which iteratively refines the LLR vector. Such iterative algorithms are known to perform well in the absence of short cycles in the graph of $\mathbf{H}$ [8]. However, the MDS property of the RS code leads to a high density of 1's in the $\mathbf{H}$ matrix and hence to short cycles. As a result, the binary matrix $\mathbf{H}$ in a standard form as in (2) is not suitable for running a message passing algorithm.

The main contribution of the adaptive parity check soft-decoding (APC-SD) algorithm of [6] is the insight that message passing algorithms will run effectively on high density parity check matrices, if cycles are eliminated within the subgraph corresponding to the low reliability received bits. The complete algorithm of [6] is summarized in Fig. 1. The key step in APC-SD is to transform the binary matrix $\mathbf{H}$ to obtain $(n-k)q$ unit weight columns corresponding to $(n-k)q$ low-reliability bits of the received vector. Even though the presentation in [6] is geared toward RS codes, the APC-SD framework is general and can be used on any linear block code. A gradient descent algorithm with computations similar to the sum-product decoding algorithm [7] is then run on the modified parity matrix. The two main steps in the gradient descent iteration are the check node update and the bit node update. The check node update computes the extrinsic information and is given by:

$$L_{ext,j}(y_i) = 2tanh^{-1}\left(\prod_{\bar{i}\in\mathcal{N}(j)\backslash i} tanh[L(y_{\bar{i}})/2]\right) \quad (3)$$

where i and j denote the bit index and the parity check index respectively. $\mathcal{N}(j)$ denotes the set of all bit indices used in the j^{th} parity check and $\mathcal{N}(j)\backslash i$ is the set of all bit indices in $\mathcal{N}(j)$ except index i. An approximation of (3), which will be used in our analysis is the min-sum formula [9] given by:

$$L_{ext,j}(y_i) = \left(\prod_{\bar{i}\in\mathcal{N}(j)\backslash i} sign[L(y_{\bar{i}})]\right)\left(min_{\bar{i}\in\mathcal{N}(j)\backslash i} L(y_{\bar{i}})\right) \quad (4)$$

From (4), we can deduce that if the number of hard-decision errors in a parity check is odd then: $\prod_{\bar{i}\in\mathcal{N}(j)\backslash i} sign[L(y_{\bar{i}})] = -sign(L(y_i))$ else $\prod_{\bar{i}\in\mathcal{N}(j)\backslash i} sign[L(y_{\bar{i}})] = sign(L(y_i))$. The bit node update step is given by:

$$L_{ext}(y_i) = \sum_{j\in\mathcal{M}(i)} L_{ext,j}(y_i)$$

where $\mathcal{M}(i)$ is the set of all parity check indices associated with bit node i. Note that this is not the *summary operator* discussed in [7], since the summation is over *all* the extrinsic messages incident on the bit node. The gradient descent step is given by:

$$L^{(j+1)}(y_i) = L^{(j)}(y_i) + \mu L_{ext}(y_i),$$

where j denotes the iteration index. This gradient descent step is shown in vector notation in Step (2)-b of Fig. 1. The parity check matrix $\mathbf{H}$ is adapted during every iteration in order to reflect the change in the LLR values.

3. ANALYSIS OF THE APC-SD ALGORITHM

In order to analyze the APC-SD algorithm, we make the following assumptions on the adapted parity check matrix:

1. The $(n-k)q$ unitary weight columns correspond to the least reliable $(n-k)q$ bits of the received vector.
2. The entries in the kq high weight columns are unbiased Bernoulli i.i.d random variables.

We denote the $(n-k)q$ bits having the least reliability in the received vector as the low reliability bits and all the other bits as high reliability bits. Let $\mathbf{I}_{hi}$ denote the set of indices of the high reliability bits and $\mathbf{I}_{lo}$ the index set of the low reliability bits. From the first assumption, the low reliability bits are associated with unit weight parity check columns. We would like to note that the first assumption is not necessarily true, since the $(n-k)q$ columns corresponding to the

low reliability bits in the original parity check matrix need not be independent. We analyze the working of the APC-SD algorithm under the following three scenarios with respect to the high reliability bits: (a) no hard-errors, (b) exactly one hard-error, (c) more than one hard-error.

3.1. Low reliability hard-decision errors

Consider the case where all the hard-decision bit errors are confined to the low reliability bits. Consequently, each parity check has either a single error or no error. If the j^{th} parity check contains an error at the bit index i, then the extrinsic information of the erroneous bit $\hat{y}_i$ in the first iteration is given by:

$$L_{ext}(y_i) = [-sign(L(y_i))]L_{min,hi} \quad i \in \mathbf{I}_{lo}, \hat{y}_i \neq c_i \quad (7)$$

where the sign is opposite of that in $L(y_i)$ since the parity check contains only one error, and

$$L_{min,hi} = min_{s \in \mathbf{I}_{hi}, s \in \mathcal{N}(j)}(|L(y_s)|).$$

Hence in the bit node update step, the extrinsic information of each erroneous bit moves the LLR toward the correct hard decision region. In addition, the gradient would be large since $L_{min,hi} > L(y_i)$ for all $i \in \mathbf{I}_{lo}$. The expectation of the bit update for the high-reliability bits can be written as:

$$\begin{aligned} \mathbf{E}\left[L_{ext}(y_i)\right] &= \mathbf{E}\left[\sum_{j \in \mathcal{M}(i)} L_{ext,j}(y_i)\right] \\ &= [sign(L(y_i))][(n-k)q/2 - e]L_{lo} \end{aligned}$$

with $i \in \mathbf{I}_{hi}, \hat{y}_i = c_i$, where e is the number of low reliability errors, and $L_{lo} = \mathbf{E}[|L(y_j)||j \in \mathbf{I}_{lo}]$. The LLR of high reliability bits increases in the correct direction if $(n-k)q/2 \geq e$.

As a result, if the number of low reliability errors are lesser than $(n-k)q/2$, both the low reliability bit LLRs and the high reliability bit LLRs are modified in the correct direction. Under such conditions, the hard-decision vector is guaranteed to converge to the correct transmitted codeword.

3.2. Single high reliability hard-decision error

We now consider the scenario of one error in the high reliability bits and e errors in the low reliability bits. For the high reliability hard-error bit, the expected number of parity checks having a single error is $(n-k)q/2 - e/2$ and the expected number of parity checks with double errors is $e/2$. Hence, the expectation of the bit node update for the erroneous high-reliability bit in the first iteration is:

$$\mathbf{E}\left[L_{ext}(y_i)\right] = [-sign(L(y_i)][(n-k)q/2 - e]L_{lo} \quad (9)$$

where $i \in \mathbf{I}_{hi}, \hat{y}_i \neq c_i$. Hence, the extrinsic information changes the LLR toward the correct decision region, if $(n-k)q/2 \geq e$. The gradient in the bit node update step is proportional to $(n-k)q/2 - e$. A bound on $P\left(\frac{e}{(n-k)q} \geq \alpha\right)$ for $\alpha \geq 0$ can be determined as follows. Using the DeMoivre-Laplace theorem [10], e can be characterized as a $\mathcal{N}(nqp, nqp(1-p))$ Gaussian random variable, where p is the probability of a bit error using BPSK transmission over the channel. Consider a channel SNR and code parameters, which result in frame error rate < 0.5 in AWGN using a bounded minimum distance decoding algorithm. Under such a condition, $p < \frac{(n-k)}{2nq}$. We then have that:

$$P\left(\frac{e}{(n-k)q} \geq \alpha\right) < \frac{1}{\sqrt{2\pi}A} exp\left(-\frac{A^2}{2}\right) \quad (10)$$

where $A = \sqrt{\frac{2(n-k)}{(1-p)}} q\left[\alpha - \frac{1}{2q}\right]$. The right hand side of (10), exponentially tends to zero for values of $\alpha > \frac{1}{2q}$. The value of the exponent is proportional to the code minimum distance and the field size exponent. It should be noted that with iterations, the gradient in (9) reduces since the number of low-reliability errors e increase. This occurs due to the LLR sign reversal of the correct low reliability bits involved in parity checks with the high reliability erroneous bit.

Based on (10), for a code over large fields, the gradient in (9) significantly reduces the absolute LLR of the high reliability hard-error bit with iterations, till it is eventually assigned to the set of low reliability bits. Under such a condition, we are back in the first case and convergence to the correct codeword is very likely.

3.3. Multiple high reliability hard-decision errors

The scenario of multiple high reliability errors is illustrated by the two high reliability bit error case. The expectation of the bit update equation for the erroneous high-reliability bits is given by:

$$\begin{aligned} \mathbf{E}\left[L_{ext}(y_i)\right] &= [-sign(L(y_i))][e/4 - (n-k)q/4 + \\ &\quad \{(n-k)q/4 - e/4\}]L_{lo} \end{aligned} \quad (11)$$

where $i \in \mathbf{I}_{hi}, \hat{y}_i \neq c_i$. The three terms in (11), involving $(n-k)q/4$ and $e/4$ represent the number of parity checks with three errors, two errors, and one error respectively. Hence, we have that for these two erroneous bits:

$$\mathbf{E}\left[L_{ext}(y_i)\right] = 0 \quad i \in \mathbf{I}_{hi}\, \hat{y}_i \neq c_i \quad (12)$$

Since the high reliability erroneous bits do not change their LLRs with iterations, the LLR of the $(n-k)q/2 - e/2$ expected low reliability correct bits, with one error in their parity check, change towards the wrong hard-decision region. All these low reliability bits can potentially be distributed among different symbols in the transmitted codeword. Thus the received vector converges to a codeword at a very large Hamming distance from the transmitted codeword in $GF(2^q)$. The analysis can be repeated for the case

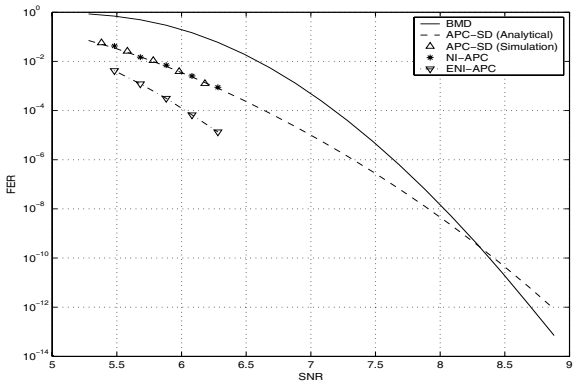

Fig. 2. Comparison of RS decoding algorithms for the $[255, 239]$ code over $GF(2^8)$

of greater than two high reliability errors where it can also be shown that the magnitude of the extrinsic information is small for the high reliability hard-error bits.

3.4. Analytical model

From the preceding discussion, it appears that irrespective of the code-parameters and the channel SNR, decoding failure is likely in the APC-SD algorithm for the case of greater than one high-reliability error. Consequently, the probability of decoding failure of the APC-SD algorithm can be expressed as:

$$\sum_{t=2}^{kq} \int_0^{\infty} P(t \text{ errors in } kq \text{ bits} | min(|LLR|) = l) f_{\mathbf{p}}(l) dl \tag{13}$$

where $\mathbf{p}$ is the $(n-k)q+1$ largest random variable in a set of nq i.i.d random variables, each of which has the density of the absolute value of the initial LLR, given by $\mathbf{z} \sim N(2/\sigma^2, 4/\sigma^2) + N(-2/\sigma^2, 4/\sigma^2)$. The density function of $\mathbf{p}$ is given by:

$$f_{\mathbf{p}}(l) = \begin{cases} B\left[F_{\mathbf{z}}(l)\right]^{(n-k)q} f_{\mathbf{z}}(l) \left[1 - F_{\mathbf{z}}(l)\right]^{(kq-1)} & l \geq 0 \\ 0 & l < 0 \end{cases}$$

where $B = \frac{(nq)!}{[(n-k)q]!(kq-1)!}$ The expression in (13) can be numerically evaluated given the channel SNR and the code parameters. The decoding performance of the APC-SD algorithm for the high rate $[255, 239]$ RS code using (13) is shown in Fig. 2. Also shown is the performance of this algorithm as obtained from simulations at moderate SNRs. It is seen that the two match well.

4. NON-ITERATIVE ADAPTIVE PARITY CHECK ALGORITHMS

Based on the preceding discussion, we now derive a non-iterative low-complexity algorithm, based on the ordered statistics decoding (OSD) concept [4], which mimics the effect of the iterative steps in the APC-SD. The pseudo-code of the Non-iterative Adaptive Parity Check (NI-APC) algorithm is given in Fig. 3. It should be noted that the NI-APC algorithm is similar to a binary form of the OSD(1) scheme described in [4].

0) Initialization:
Initialize $\mathbf{L}(\mathbf{y}) = 2/\sigma^2 \mathbf{r}$.
1) Parity Matrix Adaptation:
a) Order the LLR values in increasing reliability using absolute LLR values $|\mathbf{L}|$.
b) Reduce to unit weight, $(n-k)q$ independent columns of $\mathbf{H}$ corresponding to $(n-k)q$ low-reliability bits; thus obtain $\hat{\mathbf{H}}$.
2) Hard Decision:
$\hat{y_i} = \begin{cases} 0 & L^{(j+1)}(y_i) > 0 \\ 1 & L^{(j+1)}(y_i) \leq 0 \end{cases}$
3) Classify received vector:
$\mathbf{s} = \hat{\mathbf{H}} \cdot \hat{\mathbf{y}}$. If $wt(\mathbf{s}) < \theta$, then goto Step 4 else goto Step 5.
4) Low-reliability errors:
For all $s_j = 1$ determine $\tilde{j}$ = arg(unit weight column of $\hat{\mathbf{H}}$ with 1 in j^{th} row). Set $\hat{y}_{\tilde{j}} = \hat{y}_{\tilde{j}} \oplus 1$. Terminate.
5) Mixed errors:
a) Compute $j = argmax_{i \in \mathbf{I}_{hi}} \{\mathbf{s}' \cdot \hat{\mathbf{h}}_i\}$. Set $\hat{y}_j = \hat{y}_j \oplus 1$.
b) Compute $\mathbf{s} = \hat{\mathbf{H}} \cdot \hat{\mathbf{y}}$.
c) Goto Step 4.

Fig. 3. Pseudo-code of the Non-iterative Adaptive Parity Check (NI-APC) algorithm.

In the NI-APC algorithm, we first detect if the current received vector contains errors in the high reliability positions. This can be done by determining the weight of the bit syndrome vector obtained using the adapted parity check matrix and the hard received input vector. Since the high reliability bits have large weight columns, a single error in a high reliability position corresponds to a weight of about $(n-k)q/2$ in the syndrome vector. In contrast, a low reliability bit error only leads to a unit increase in the syndrome vector weight. The distribution of the syndrome vector weight for the $[255, 239]$ RS code over $GF(2^8)$ is shown in Fig. 4. The syndrome vector weight density can also be computed analytically to set an optimal threshold θ in Step 3 of the NI-APC algorithm. In the case of high reliability hard-errors, the bit corresponding to the column of $\mathbf{H}$, which has the highest correlation with the syndrome vector is declared to be in error and its hard-decision is flipped.

The simulation performance of the NI-APC algorithm is shown in Fig. 2. It is seen that it closely matches the performance of the APC-SD algorithm. The NI-APC algorithm has lower complexity compared to the APC-SD algorithm since all iterations involving soft-values have been eliminated. It should be noted that NI-APC will not match the APC-SD performance for all codes and for all SNR values. In particular, the FER of the NI-APC will increase at an SNR and for code parameters at which the two modes in Fig. 4 are not well separated. For such cases, decoded vector likelihoods, should augment the correlation metric to

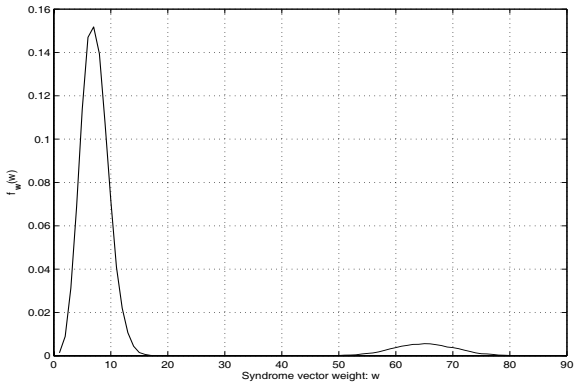

Fig. 4. Density of the syndrome vector weight for the $[255, 239]$ code over $GF(2^8)$ at $SNR = 6.0\ dB$.

5) Mixed errors:

a) Compute $\mathbf{c}orr_1 = max_{i \in \mathbf{I}_{hi}}\{\mathbf{s}' \cdot \hat{\mathbf{h}}_i\}$. Let $\tilde{j}$ be the argument.
b) If $(wt(\mathbf{s}) - \mathbf{c}orr_1) < \gamma$, then:
  i] $\hat{y}_{\tilde{j}} = \hat{y}_{\tilde{j}} \oplus 1$.
  ii] Compute $\mathbf{s} = \hat{\mathbf{H}} \cdot \hat{\mathbf{y}}$.
  iii] For all $s_j = 1$ determine $\tilde{j}$ = arg(unit weight column of $\hat{\mathbf{H}}$ with 1 in j^{th} row). Set $\hat{y}_{\tilde{j}} = \hat{y}_{\tilde{j}} \oplus 1$. Terminate.
c) Else:
  i] Compute $\mathbf{c}orr_2 = max_{i,j \in \mathbf{I}_{hi}}\{\mathbf{s}' \cdot (\hat{\mathbf{h}}_i \oplus \hat{\mathbf{h}}_j)\}$.
    Let (j_1, j_2) be the arguments.
  ii] If $\mathbf{c}orr_2 - \mathbf{c}orr_1 < \delta$ goto Step 5 (b)-[i].
  iii] $\hat{y}_{j1} = \hat{y}_{j1} \oplus 1$, $\hat{y}_{j2} = \hat{y}_{j2} \oplus 1$.
  iv] Goto Step 5 (b)-[ii].

Fig. 5. Pseudo-code of the Extended Non-iterative Adaptive Parity Check (ENI-APC) algorithm.

classify the received vector (Step 3) and to determine the high reliability bit in error (Step 5-(a)).

We now propose an extension to deal with the two-error case to improve the performance of the non-iterative algorithm. This extension is similar to the OSD(2) scheme of [4]. The pseudo-code of the Extended Non-iterative Adaptive Parity Check (ENI-APC) algorithm is given in Fig. 5. The first four steps are identical to the NI-APC algorithm of Fig. 3. The modification proposed pertains to Step 5, which detects and corrects two high reliability hard-errors.

The main additional complexity is in Step (5)-(c)[i], which computes correlations of the syndrome vector with the XOR of two columns of the adapted parity check matrix. A straight forward implementation of this procedure requires $kq \times (kq + 1)/2 \times (n - k)q$ XOR operations and an equal amount of AND operations. This is $\frac{(k/n)^2}{1-(k/n)}$ times the complexity of generating the adapted parity check matrix. For the $[255, 239]$ RS code, this ratio is around 14. However, since the algorithm is not iterative, we save on the repeated parity matrix adaptations of the APC-SD algorithm. In addition, the two-error testing branch is taken with a low probability which varies inversely with SNR. The performance of the ENI-APC algorithm is shown in Fig. 2.

5. CONCLUSIONS

We have presented an expectation based analysis of the decoding performance of the adaptive parity check soft-decision decoding algorithm. Based on this analysis, we have also proposed two non-iterative algorithms using adaptive parity check matrices that meet and extend the performance of the original algorithm for the important case of the high rate $[255, 239]$ RS code.

6. ACKNOWLEDGMENTS

The authors gratefully acknowledge Jing Jiang and Prof. Krishna R. Narayanan of Texas A & M University for providing an extended version of [6].

7. REFERENCES

[1] R. E. Blahut, *Algebraic Codes for Data Transmission*, Cambridge,UK: Cambridge University Press, 2002.

[2] G. D. Forney Jr., "Generalized minimum distance decoding," *IEEE Trans. on Information Theory,* vol. 12, pp. 125–131, Apr. 1966.

[3] D. Chase, "Class of algorithms for decoding block codes with channel measurement information," in *IEEE Trans. on Information Theory,* vol. 18, pp. 170–182, Jan. 1972.

[4] M. P. C. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. on Information Theory,* vol. 41, pp. 1379–1396, Sept. 1995.

[5] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. on Information Theory,* vol. 49, pp. 2809–2825, Nov. 2003.

[6] J. Jiang and K. R. Narayanan, "Iterative soft decision decoding of Reed Solomon codes based on adaptive parity check matrices," *Proc. of the IEEE Intl. Symp. on Information Theory,* July 2004.

[7] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Information Theory,* vol. 47, pp. 498–519, Feb. 2001.

[8] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. on Information Theory,* vol. 47, pp.619–637, Feb. 2001.

[9] J. Li, K. R. Narayanan, and C. N. Georghiades, "An efficient decoding algorithm for cycle-free convolutional codes and its applications," in *Proc. of IEEE Global Telecom. Conf.*, vol. 2, pp. 1063–1067, Nov. 2001.

[10] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd Edition, New York, McGraw-Hill, 1991.