Iterative Soft-Input Soft-Output Bit-Level Reed-Solomon Decoder Based on Information Set Decoding

Yuval Genga, Olutayo O. Oyerinde, *Senior Member, IEEE,* and Jaco Versfeld, *Senior Member, IEEE*

***Abstract*—In this paper, a bit-level decoder is presented for soft-input soft-output iterative decoding of Reed-Solomon (RS) codes. The main aim for the development of the proposed algorithm is to reduce the complexity of the decoding process, while yielding a relatively good error correction performance, for the efficient use of RS codes. The decoder utilises information set decoding techniques to reduce the computational complexity cost by lowering the iterative convergence rate during the decoding process. As opposed to most iterative bit-level soft-decision decoders for RS codes, the proposed algorithm is also able to avoid the use of belief propagation in the iterative decoding of the soft bit information, which also contributes to the reduction in the computational complexity cost of the decoding process. The performance of the proposed decoder is investigated when applied to short RS codes. The error correction simulations show the proposed algorithm is able to yield a similar performance to that of the Adaptive Belief Propagation (ABP) algorithm, while being a less complex decoder.**

***Index Terms*—Reed-Solomon codes, Bit-level decoding, Iterative decoding, Soft-decision decoding, Information set decoding, Decoding complexity.**

I. Introduction

REED-Solomon (RS) codes belong to a class of high performing error correction codes that were developed by Irving S.Reed and Gustave Solomon [1]. Due to the strong algebraic properties and good error detection and correction performance, a lot of research has been applied to the development of high performing decoding schemes for this class of codes.

Soft-decision decoders for RS codes working in the field, specifically the ones based in the Galois Field GF(2^b) (where b is any positive integer), can be divided into two categories. They can either be symbol-level decoders or bit-level decoders. Most soft-decision decoders for RS codes work on a symbol-level. Examples of symbol-level decoders include the widely used Koetter and Vardy (KV) algorithm [2] and the parity check transformation algorithm (PTA) [3] [4]. Previous research works on soft-decision decoders have shown gains in terms of error correcting performance attained when working on a bit-level compared to the symbol-level [5], [6], [7], [8], [9]. Based on this, implementation of bit-level soft-decision decoding algorithms for RS codes has been an area of active research for a long time [5], [6], [7], [10], [11], [12], [13], [14], [15].

The adaptive belief propagation (ABP) algorithm [5] is a widely used bit-level decoder for linear block codes due to its good error correcting capability [5], [6], [12], [13], [16]. The ABP algorithm was devised with the aim of applying belief propagation techniques on codes defined by a parity check matrix with a dense structure [5]. The belief propagation decoding algorithm is designed to take advantage of the sparse structure presented by the parity check matrix of a Low Density Parity Check (LDPC) code so as to decode the received vector efficiently [17]. Due to the dense nature of RS code, a binary image expansion [18] is performed on the parity check matrix ($\boldsymbol{H}$) so as to make it sparse. This is done to enable the use of the belief propagation algorithm during the decoding process. The generic form of the ABP decoder presented in [5] works by first converting the bit reliabilities of the codeword into their corresponding log likelihood ratios (LLR). The binary image form of the $\boldsymbol{H}$ is then adapted based on these LLR values using Gaussian row reduction techniques. The belief propagation is then applied to the adapted binary image of the $\boldsymbol{H}$ matrix so as to decode the received vector. In [5], the ABP has been shown to yield a significant gain when compared to widely used RS decoders including the KV algorithm, Berlekamp-Massey (BM) algorithm and the Algebraic hard-decision decoder. However, the gains achieved by the ABP algorithm come at the cost of a high computational complexity [5], [13]. Further modifications have been made to the generic form of the decoder [5], [6], [12], [13], [14], [16], [19]. However, these changes either add to or do not significantly reduce the complexity cost of iteratively applying belief propagation during decoding [5], [13], [20].

The parity check transformation algorithm (PTA) [4] is a symbol level RS decoder that, just like the ABP, utilises row reduction techniques on the $\boldsymbol{H}$ matrix based on the symbol reliability. The PTA works by sorting the maximum symbol reliabilities obtained from each column of a reliability matrix and using them to transform the $\boldsymbol{H}$ matrix. The $\boldsymbol{H}$ matrix is transformed using the row reduction technique shown in [21] [22] to match the corresponding reliability information with the columns of $\boldsymbol{H}$. Once the $\boldsymbol{H}$ matrix is transformed, the reliabilities are then corrected based on the values of the

This work was supported in part by the National Research Foundation (NRF) South Africa. The financial support of the Centre for Telecommunications Access and Services (CeTAS), the University of the Witwatersrand, Johannesburg, South Africa is also acknowledged.

Y. Genga and O. O. Oyerinde are with the School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, South Africa. (e-mail: yuval.genga@wits.ac.za and olutayo.oyerinde@wits.ac.za).

J. Versfeld is with the Department of Electrical and Electronic Engineering, Stellenbosch University, Matieland, South Africa, South Africa. (e-mail: djjversfeld@sun.ac.za).

syndrome vector obtained from the scalar product of the hard decision vector and each row of the transformed $\boldsymbol{H}$ matrix. The PTA has been shown to outperform the KV decoder and the BM algorithm, in terms of error correction performance [3] [4]. In [9], the PTA was modified to allow for a bit-level implementation. The bit-level PTA is shown in [9] to yield a comparable error correction performance to the ABP algorithm for a (15,7) and a (15,11) RS code transmitted using a 16-QAM modulation scheme. The PTA is able to achieve gains in terms of error correction, in both symbol and bit-level, at the cost of running numerous iterations.

Information set decoding (ISD) was first presented by Prange [23] for the decoding of Cyclic codes. Since then, the algorithm has had different modifications for application in error correction of linear block codes [24], [25], [26], [27] and Cryptography [28], [29]. In all its forms, ISD uses k linearly independent bits or symbols from the received vector to re-encode a unique codeword during the decoding process [24]. The k symbols or bits used to obtain the unique codeword are referred to as the information set. The most widely used implementations of ISD in the error correction of linear block codes have been applied to binary codes. ISD algorithms often utilise ordered statistics decoding (OSD) and list decoding techniques to obtain the decoded vector of the binary code [25], [30], [31], [32]. In [27] a Field Programmable Gate Array (FPGA) implementation for ISD is presented when applied to RS codes. Besides the FPGA implementation, part of the novelty of the approach proposed in [27] is that ISD is applied to a class of nonbinary codes. However it is important to note that the decoding method presented in [27] is not applied at the symbol-level, but instead implemented at a bit-level. Similar to the ABP, the decoder is able to work at a bit-level by performing a binary image expansion on the $\boldsymbol{H}$ matrix and converting the RS code into its bit form. The proposed decoding method in [27] is based on a modified OSD implementation of the ISD technique used in [33]. The list of candidate codewords from the OSD are obtained using order-1 reprocessing [30] to apply bit flipping based decoding on the syndrome weights from the list of subvectors. The subvectors are then used in the re-encoding of the candidate codewords. ISD decoders have the advantage of being generally less complex than other decoding techniques [24] [34]. However, the use of OSD and list decoding techniques ensure ISD decoders that utilise soft information are soft-input hard-output algorithms. This prohibits the use of such decoding algorithms for situations where the RS decoders are required to make full use of bit based soft information, like in the case of satellite transmission[10] or for iterative decoding [35].

Motivation and Objectives

Soft-decision decoders used for RS codes give a good error correction performance, but at the cost of some form of complexity. The ABP algorithm gives a significant gain compared to hard decision decoders and the widely used KV algorithm, but this comes at the cost of a higher computational complexity [5][6]. The PTA has also been shown to be a good decoder while outperforming the KV algorithm in terms of symbol error rate [4]. This is at the cost of the algorithm running numerous iterations [36], therefore making the PTA a computationally intensive algorithm. The 'algorithm complexity vs error correction performance' tradeoff is quite common in the field of telecommunications when it comes to the selection of appropriate decoding algorithms. The main objective of this research is the presentation of a decoding technique that iteratively utilizes the soft information outputted from the noisy channel to yield a high error correction performance at a reduced computational complexity cost.

Work in this paper focuses on the development of a low complexity iterative soft-input soft-output decoding approach for RS codes that is able to yield a good error correction performance.

The proposed decoding approach is able to take advantage of the reduced complexity that comes with an ISD implementation [24] in the error correction process. Except for the bit-level PTA, all iterative soft-input soft-output decoders for RS codes are in some way a modification of the ABP decoder. In addition to the development of an iterative soft-input soft-output ISD based bit-level decoder, part of the novelty of the decoding technique presented in this research is the implementation of a bit-level iterative soft-decision decoder for RS codes without the use of the belief propagation algorithm. Implementation of belief propagation is avoided in the proposed decoding technique due to the high computational complexity cost presented when using the ABP algorithm [5], [11], [13], [37]. The decoder proposed in this research is a message passing algorithm, that takes advantage of the sparse structure presented by the binary image of the $\boldsymbol{H}$ matrix to iteratively decode the received vector. ISD is implemented, as part of the stopping criteria in the proposed iterative bit-level decoding approach, with the aim of improving the iterative convergence rate during the decoding process. An improved iterative convergence rate contributes to a reduction in the computational complexity cost as it reduces the total number of operations carried out during the entire decoding process of the received vector.

The rest of the paper is structured as follows: a detailed description of how the proposed decoder works is given in Section III, thereafter an analysis of the proposed algorithm and simulation results are presented in Section IV, then a complexity analysis of the proposed decoder is investigated in Section V and Section VI gives the conclusion to the findings obtained in this paper.

II. Information Set decoding: Importance of the Systematic structure to the decoding process

Assume a (n, k) RS code $\mathcal{C}$, in the field GF(2^b), is defined by a systematic parity check matrix $\boldsymbol{H}$ having the dimensions $m \times n$, where $m = (n - k)$.

$$\boldsymbol{H} = [\boldsymbol{I}\ \boldsymbol{Q}], \tag{1}$$

where $I = m \times m$ identity submatrix and $Q = (n - m) \times m$ parity submatrix. The syndrome $S = 0$ can be found by multiplying a valid codeword c from $\mathcal{C}$

$$S = c \times \boldsymbol{H}^{\top}, \tag{2}$$

where $\boldsymbol{H}^\top$ is the transpose of the $\boldsymbol{H}$ matrix. We can rewrite (2) in the form shown in (3)

$$S = (c_I \times \boldsymbol{I}^\top) + (c_Q \times \boldsymbol{Q}^\top) \tag{3}$$

where the c_I and c_Q represent subvectors of the codeword c that correspond to the identity and parity submatrix respectively. As a result of the syndrome $S = 0$, the additive inverse property of the field GF(2^b) [38] can be applied to (3) to give

$$(c_I \times \boldsymbol{I}^\top) = (c_Q \times \boldsymbol{Q}^\top) \tag{4}$$

Based on the systematic structure of $\boldsymbol{I}^\top$, we can further reduce (4) to (5)

$$c_I = (c_Q \times \boldsymbol{Q}^\top) \tag{5}$$

From (5), it can be seen the subvector c_I of the codeword c can be correctly obtained using the the subvector c_Q. This is the information set decoding principle behind the proposed decoding approach presented in this research.

III. The Proposed Decoding Scheme

To understand how the proposed decoder works, the relevant notation is first established. The decoder works on a bit-level, this means a binary image expansion is performed on $\boldsymbol{H}$. In what follows, the binary image expansion of the $\boldsymbol{H}$ matrix for the RS code is obtained as described. Each element α^z of the field GF(2^b), where $0 \leq z \leq (2^b - 2)$, is replaced by a corresponding $b \times b$ binary matrix $\boldsymbol{B}^z$, where $\boldsymbol{B}$ is the companion matrix of a primitive polynomial which creates the field $\mathbb{F}_{2^b}$ [16][18]. The binary image expansion of $\boldsymbol{H}$ is denoted as $\boldsymbol{\mathcal{H}}$ and has the dimensions $M \times N$, where $M = (nb - kb)$ and $N = nb$.

We now consider an RS codeword c in the extension field. The codeword c is transmitted using a selected modulation scheme through a noisy channel. The soft information received at the output of the channel is then used to create the reliability matrix $\boldsymbol{\beta}$. The matrix $\boldsymbol{\beta}$ has the dimensions $2 \times N$, where the 1st and 2nd row of $\boldsymbol{\beta}$ represent the reliability of the selected bit index being a "0" and a "1" respectively.

The techniques used to create $\boldsymbol{\beta}$ depend on the channel and the selected modulation scheme. For instance when transmitting through an Additive White Gaussian Noise (AWGN) channel using a Binary Phase Shift Keying (BPSK) modulation scheme, the codeword c is first converted into its bit form by representing each element α^z with the respective binary polynomial $a_0 + a_1\alpha + ... + a_{b-1}\alpha^{(b-1)}$. The binary codeword is denoted as cb, and has the length nb. After transmission, the vector rb is received at the output of the noisy channel. The soft information in the vector rb is then used to find the bit reliabilities of either being a "0" or a "1" using the method presented in [39][40]. When the symbols of the codeword c are transmitted using a 16-QAM modulation scheme with Gray mapping through an AWGN channel, the reliabilities used to fill the rows in the matrix $\boldsymbol{\beta}$ are obtained using the technique presented in [41] from the soft information received at the output of the channel.

Once the matrix $\boldsymbol{\beta}$ is created, each of the columns is then scaled using the approach presented in [9]. This is carried out to ensure that the reliabilities in each column of $\boldsymbol{\beta}$ add up to 1 before being fed into the decoder.

For each iteration of the proposed algorithm, the decoding process can be summarised by the following steps:

1) *Finding the maximum bit reliabilities:* The maximum reliabilities in each column of $\boldsymbol{\beta}$ are identified and arranged in the vector A as shown in (6)

$$\boldsymbol{A} = [A_1, A_2, ..., A_N] \tag{6}$$

where $A_j = \text{argmax}(\beta_j)$, $0 \leq j \leq N$ and β_j represents each column of the matrix $\boldsymbol{\beta}$. These reliabilities are then sorted in ascending order. The original indices of the reliabilities are identified as well and stored in terms of their ascending order in the vector Y

$$\boldsymbol{Y} = [\ Y_1\ ,\ Y_2\ ,\ ...\ ,\ Y_N\ ]. \tag{7}$$

For $K = kb$, the K highest values of the vector A are considered to be the most reliable. An approach similar to the information set decoding technique presented in [27] is then applied. This ensures, after row reduction is performed, the indices of the K highest reliabilities match the parity submatrix of the now quasi-systematic structure of $\boldsymbol{\mathcal{H}}$. For ease of notation, the matrix obtained from the row reduction of the rearranged columns of $\boldsymbol{\mathcal{H}}$ is represented as $\boldsymbol{\mathcal{H}}^\perp$.

It is important to note that the matrix $\boldsymbol{\mathcal{H}}$ has a full row rank. This means that there exists a total of M independent columns present in $\boldsymbol{\mathcal{H}}$. However, as noted in [5], there is no guarantee that the M least reliable indices found in $\mathbf{Y}$ will match these columns during row reduction. This means that not all the most reliable K indices will match the parity submatrix for every row reduction operation performed on $\boldsymbol{\mathcal{H}}$. When this happens, any M indexes that matches an identity submatrix of $\boldsymbol{\mathcal{H}}^\perp$ are considered to be unreliable, and any K bit indexes that match the parity submatrix of $\boldsymbol{\mathcal{H}}^\perp$ are considered to be reliable.

2) *Hard-decision detection and the Syndrome check:* Hard-decision detection, similar to [4], is then performed on $\boldsymbol{\beta}$ to obtain the vector $\widehat{\boldsymbol{cb}}$. The syndrome is then calculated by getting the scalar product of the vector $\widehat{\boldsymbol{cb}}$ and each row of $\boldsymbol{\mathcal{H}}^\perp$ as shown in (8)

$$\boldsymbol{S}_i = \widehat{\boldsymbol{cb}} \cdot \boldsymbol{\mathcal{H}}^\perp_i, \tag{8}$$

where $1 \leq i \leq M$ is used to denote each row of $\boldsymbol{\mathcal{H}}^\perp$ and each value of the syndrome vector, $\boldsymbol{S}$. Due to the decoder working on a bit-level, the syndrome calculation in (8) can be rewritten in the form shown in (9)

$$\boldsymbol{S}_i = \sum \widehat{\boldsymbol{cb}}_{t_i}, \tag{9}$$

where $\boldsymbol{t}_i$ represents all the indices of the participating bits of $\widehat{\boldsymbol{cb}}$ in the i^{th} syndrome check equation and is expressed as

$$\boldsymbol{t} = \{N : \boldsymbol{\mathcal{H}}^\perp_{i,N} = 1\}. \tag{10}$$

3) *Obtaining the votes:* During the syndrome calculation, votes are cast for each bit. Each bit gets a vote of either

being a "0" or a "1" based on the extrinsic information. This means, for each row, all the participating bits except the one being investigated take part in the vote. Based on the vote, The syndrome calculation in (9) is rewritten as

$$S_i = \widehat{cb}_y + \sum \widehat{\boldsymbol{cb}_{\boldsymbol{t}'}}, \tag{11}$$

where $y \in \boldsymbol{t}_i$ represents the index of the bit being voted for. The subvector $\boldsymbol{t}'$ represents the set of indices in t_i without the bit index y. Assuming the set of bits $\widehat{\boldsymbol{cb}_{\boldsymbol{t}'}}$ are all correct and $S_i = 0$, the calculation of votes can be derived from the additive inverse property of the field GF(2^b) [38] and is represented as

$$\widehat{cb}_y = \sum \widehat{\boldsymbol{cb}_{\boldsymbol{t}'}}. \tag{12}$$

From (12), $\widehat{cb}_y$ is found to be either a "0" or "1". This counts as a vote for the bit $\widehat{cb}_y$. This process is repeated for all participating bits of $\widehat{\boldsymbol{cb}}$ in every row of $\boldsymbol{\mathcal{H}}^{\perp}$, with the votes being stored in the matrix $\boldsymbol{V}$ as seen in (13)

$$\boldsymbol{V} = \begin{bmatrix} V_{0,1}, & V_{0,2}, & \dots & V_{0,N} \\ V_{1,1}, & V_{1,2}, & \dots & V_{1,N} \end{bmatrix}, \tag{13}$$

where $V_{0,j}$ and $V_{1,j}$ represents the total votes each bit index gets for being a "0" and a "1" respectively, for all the rows of $\boldsymbol{\mathcal{H}}^{\perp}$,

4) *Obtaining the confidence rating:* The first step to obtaining the confidence rating of each bit is to get the voting ratios, $\boldsymbol{\vartheta}$, as shown in (14)

$$\boldsymbol{\vartheta} = \begin{bmatrix} \frac{V_{0,1}}{V_{T,0}}, & \frac{V_{0,1}}{V_{T,1}}, & \dots & \frac{V_{0,N}}{V_{T,N}} \\ \frac{V_{1,1}}{V_{T,1}}, & \frac{V_{1,2}}{V_{T,2}}, & \dots & \frac{V_{1,N}}{V_{T,N}} \end{bmatrix}, \tag{14}$$

where each $V_{T,j}$ represents the total number of votes each bit index gets and it is computed as

$$V_{T,j} = V_{0,j} + V_{1,j}. \tag{15}$$

The confidence rating, $\boldsymbol{\Gamma}$, for each bit is then calculated by dividing the voting ratios in $\boldsymbol{\vartheta}$ by a value of ρ as shown in (16)

$$\boldsymbol{\Gamma} = \frac{\boldsymbol{\vartheta}}{\rho}, \tag{16}$$

where ρ represents the divisor which is a constant predefined value input during the initialisation of the algorithm. The values in the matrix Γ can be represented as

$$\boldsymbol{\Gamma} = \begin{bmatrix} \Gamma_{0,1}, & \Gamma_{0,2}, & \dots & \Gamma_{0,N} \\ \Gamma_{1,0}, & \Gamma_{1,2}, & \dots & \Gamma_{1,N} \end{bmatrix}, \tag{17}$$

5) *Updating $\boldsymbol{\beta}$:* The values in $\boldsymbol{\Gamma}$ represent the level of confidence that a bit is correct and should be updated in $\boldsymbol{\beta}$. The update works by adding the indexed confidence ratio to the corresponding reliability in $\boldsymbol{\beta}$, based on the vote in (12) for being either a "0" or "1". The update can be summarized as follows

$$\boldsymbol{\beta}^{(f+1)}_{(\widehat{cb}_y, j)} = \boldsymbol{\beta}^{f}_{(\widehat{cb}_y, j)} + \boldsymbol{\Gamma}_{(\widehat{cb}_y, j)}, \tag{18}$$

where f is used to denote the current iteration number of the decoder. The value of $\widehat{cb}_y$ is either a "0" or a "1". This value is used to identify which column in $\boldsymbol{\beta}$, with the reliability indexed by j, to update with the corresponding confidence rating. The notation y represents the index of the bit considered during (12). All the reliabilities in $\boldsymbol{\beta}$ are updated based on their participation in each row of the matrix $\boldsymbol{\mathcal{H}}^{\perp}$. After all the N indices are updated, $\boldsymbol{\beta}^{(f+1)}$ is scaled to ensure all the columns add up to 1 in preparation for the next iteration.

A. The Decoding Condition and Thresholds

The proposed algorithm works iteratively and has two stopping conditions. The first is when $\boldsymbol{S} = 0$. The second is when the decoding condition is met. The decoding condition is based on ISD. That is, it makes use of a set of K bits to re-encode the decoded codeword based on the rearranged systematic structure of $\boldsymbol{\mathcal{H}}^{\perp}$.

The decoding condition is met if the algorithm can ascertain that the information set of K bits with indices matching the parity submatrix is correct. If the information set is determined to be correct, then the remaining M bits that match the identity matrix can be decoded.

In order to understand how this works, consider the syndrome equation between $\widehat{\boldsymbol{cb}}$ and the rearranged systematic matrix $\boldsymbol{\mathcal{H}}^{\perp}$ in the form shown in (19).

$$\boldsymbol{S} = (\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_M} \times \boldsymbol{I}^{\top}_{\boldsymbol{\mathcal{H}}^{\perp}}) + (\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K} \times \boldsymbol{Q}^{\top}_{\boldsymbol{\mathcal{H}}^{\perp}}), \tag{19}$$

where $\boldsymbol{I}^{\top}_{\boldsymbol{\mathcal{H}}^{\perp}}$ and $\boldsymbol{Q}^{\top}_{\boldsymbol{\mathcal{H}}^{\perp}}$ match the transpose of the column indices of identity and parity submatrices of $\boldsymbol{\mathcal{H}}^{\perp}$, while $\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_M}$ and $\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K}$ match the indices of $\widehat{\boldsymbol{cb}}$ that correspond to the columns of the identity and parity submatrices of $\boldsymbol{\mathcal{H}}^{\perp}$ respectively. Due to $\boldsymbol{I}_{\boldsymbol{\mathcal{H}}^{\perp}}$ being an identity matrix, (19) can be reduced further as shown in (20).

$$\boldsymbol{S} = \widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_M} + (\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K} \times \boldsymbol{Q}^{\top}_{\boldsymbol{\mathcal{H}}^{\perp}}), \tag{20}$$

If the $\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K}$ indices are correct, the bit values of $\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_M}$ can be obtained by assuming $\boldsymbol{S} = 0$ and applying the additive inverse property of the field GF(2^b) to give

$$\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_M} = \widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K} \times \boldsymbol{Q}^{\top}_{\boldsymbol{\mathcal{H}}^{\perp}}. \tag{21}$$

Similar to (12), the bit-level expression in (21) can be reduced further by considering only the indices of the participating bits to give

$$\widehat{cb}_{Y_{M_i}} = \sum \widehat{\boldsymbol{cb}}_{\boldsymbol{t}_{K_i}} \tag{22}$$

where $\widehat{cb}_{Y_{M_i}}$ and $\widehat{\boldsymbol{cb}}_{\boldsymbol{t}_{K_i}}$ are the participating bits of $\widehat{\boldsymbol{cb}}_{Y_M}$ and $\widehat{\boldsymbol{cb}}_{Y_K}$ in the i^{th} row of $\boldsymbol{\mathcal{H}}^{\perp}$ respectively.

The algorithm is able to determine if the bits in the subvector $\widehat{\boldsymbol{cb}}_{Y_K}$, referred to as the information set, are correct by setting a threshold. The threshold, τ, is defined as the minimum number of syndrome check equations each of the K bits in the information set should satisfy for (22) to be applied in the decoding of the received vector. The higher the value of τ, the more confidence the algorithm has that the most reliable

K indices are correct. However, this comes at the expense of the algorithm running more iterations.

The main advantage of using the decoding condition as a stopping criteria is that the algorithm is able to use K bits to decode an entire received vector of length N. This assists in reducing the number of iterations required to decode the received vector, because the algorithm does not have to confirm if every single bit is correct before it can break the iterative decoding process.

A detailed summary of the proposed decoding approach is presented in Algorithm 1. Also, a flow diagram that follows the stages involved in the decoding process is represented in Fig. 1. For purpose of notation in the summaries presented in Algorithm 1 and Fig. 1, the decoded vector is denoted using $\widehat{\boldsymbol{C}}$ and $\boldsymbol{S}_{\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K}}$ is used to represent the minimum number of syndrome checks satisfied by the each of the K bits in the vector $\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K}$.

Algorithm 1: Summary of the proposed Bit-level Decoder.

Input: The received vector, $\boldsymbol{r}$
Output: The decoded vector $\widehat{\boldsymbol{C}}$.

Initialize: Obtain $\boldsymbol{\beta}$ from $\boldsymbol{r}$. Set the values of τ and ρ.
repeat
- ***•Obtaining the Informational set***:
- Obtain the vector $\boldsymbol{A}$ from $\boldsymbol{\beta}$.
- Row reduce $\boldsymbol{\mathcal{H}}$ based on the sorted reliability indexes stored in vector $\boldsymbol{Y}$ to form the matrix $\boldsymbol{\mathcal{H}}^{\perp}$.
- ***•Syndrome check and Decoding condition***:
- Hard-decision detection is applied to $\boldsymbol{\beta}$ to obtain $\widehat{\boldsymbol{cb}}$.
- Calculate the syndrome $\boldsymbol{S}$ using (9) and note the values of $\boldsymbol{S}_{\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K}}$
- ***•Voting for the bits***
- Apply (12) for the participating bits in each row of $\boldsymbol{\mathcal{H}}^{\perp}$ and store the vote tally for each bit in $\mathbf{V}$.
- ***•Obtaining the bit Confidence ratings***
- Obtain the voting ratios, $\boldsymbol{\vartheta}$, using (14) and (15).
- Apply (16) to get the confidence rating for each bit and store the values in the vector $\boldsymbol{\Gamma}$.
- ***•Update*** $\boldsymbol{\beta}$
- Update $\boldsymbol{\beta}$ using the respective confidence rating as shown in (18)

until $\boldsymbol{S} = 0$ or $\boldsymbol{S}_{\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K}} = \tau$
if $\boldsymbol{S}_{\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K}} = \tau$ **then**
- compute (22) to re-encode the decoded vector $\widehat{\boldsymbol{C}}$ from the information set $\widehat{\boldsymbol{cb}}_{\boldsymbol{Y}_K}$.

else
- $\widehat{\boldsymbol{C}} = \widehat{\boldsymbol{cb}}$

IV. Results and Analysis

In this section, simulation results for the proposed iterative decoder and its variants are presented. For ease of notation, the proposed decoder is referred to as the k Bit Decoding algorithm and is denoted as the kBD algorithm.

A. Analysis of the Proposed Bit-Level Decoding Algorithm

Simulations are run to find the optimum performance conditions for the kBD algorithm using different values of τ and ρ. The performance of the kBD algorithm using different values of τ is first investigated. A nearly half rate $(15, 7)$ RS code is used in this simulation, with the symbols being transmitted through an AWGN channel using a 16-QAM modulation scheme with Gray mapping. The value of $\rho = 50$ is used for these simulations. Results for the simulations are measured in terms of Bit Error Rate (BER) and the average number of iterations. These results are presented in Fig. 2 and Fig. 3 respectively.

From Fig. 2 it can be seen that the error correction performance of the kBD algorithm works best when $\tau \geq 7$. It can also be seen from Fig. 3 that working with $\tau = 7$ presents a more efficient kBD algorithm. This is because it requires less than half of the average number of iterations used by kBD algorithm with $\tau = 10$ during the decoding process, to achieve a similar BER performance.

Tests to determine the optimum value of ρ are also carried out. Similar conditions are used for transmission of the $(15, 7)$ RS code. The kBD algorithm is implemented with a $\tau = 7$. The results for these simulations are presented in Fig. 4 and Fig. 5.

The kBD algorithm with values of $\rho \geq 50$ are able to achieve a slightly higher gain in terms of BER when compared to the kBD algorithm with values of $\rho \leq 30$ as seen in Fig. 4. The kBD algorithm with $\rho = 50$ is shown to be an efficient decoder as it requires less iterations to yield a comparable BER performance to the other versions of the algorithm, with values of $\rho > 50$, as seen in Fig. 5. Based on this, the kBD algorithm with $\rho = 50$ and $\tau = 7$ is used to benchmark the performance of the proposed algorithm with other iterative soft-input soft-output bit-level decoders.

B. Performance Analysis of the Proposed Bit-Level Decoding Algorithm

Half rate codes

In this section the performance of the kBD algorithm is benchmarked against the bit-level implementation of the PTA and the ABP algorithm. Simulations are run on a nearly half rate $(15, 7)$ RS code. The ABP is simulated with the value of $\alpha = 0.05$ and is set to run with a maximum number of 20 iterations [5]. The bit-level implementation of the PTA, denoted as PTA_{bl}, is run with a value of $\delta = 0.01$ [9].

For these simulations, a version of the kBD algorithm with a lower computational complexity cost than the original implementation is presented. This version of the kBD algorithm only performs Gaussian row reduction once on $\boldsymbol{\mathcal{H}}$, in the first iteration, to obtain $\boldsymbol{\mathcal{H}}^{\perp}$. The same $\boldsymbol{\mathcal{H}}^{\perp}$ is then used throughout the entire iterative decoding process of the received vector. This version of the kBD algorithm is referred to as the none

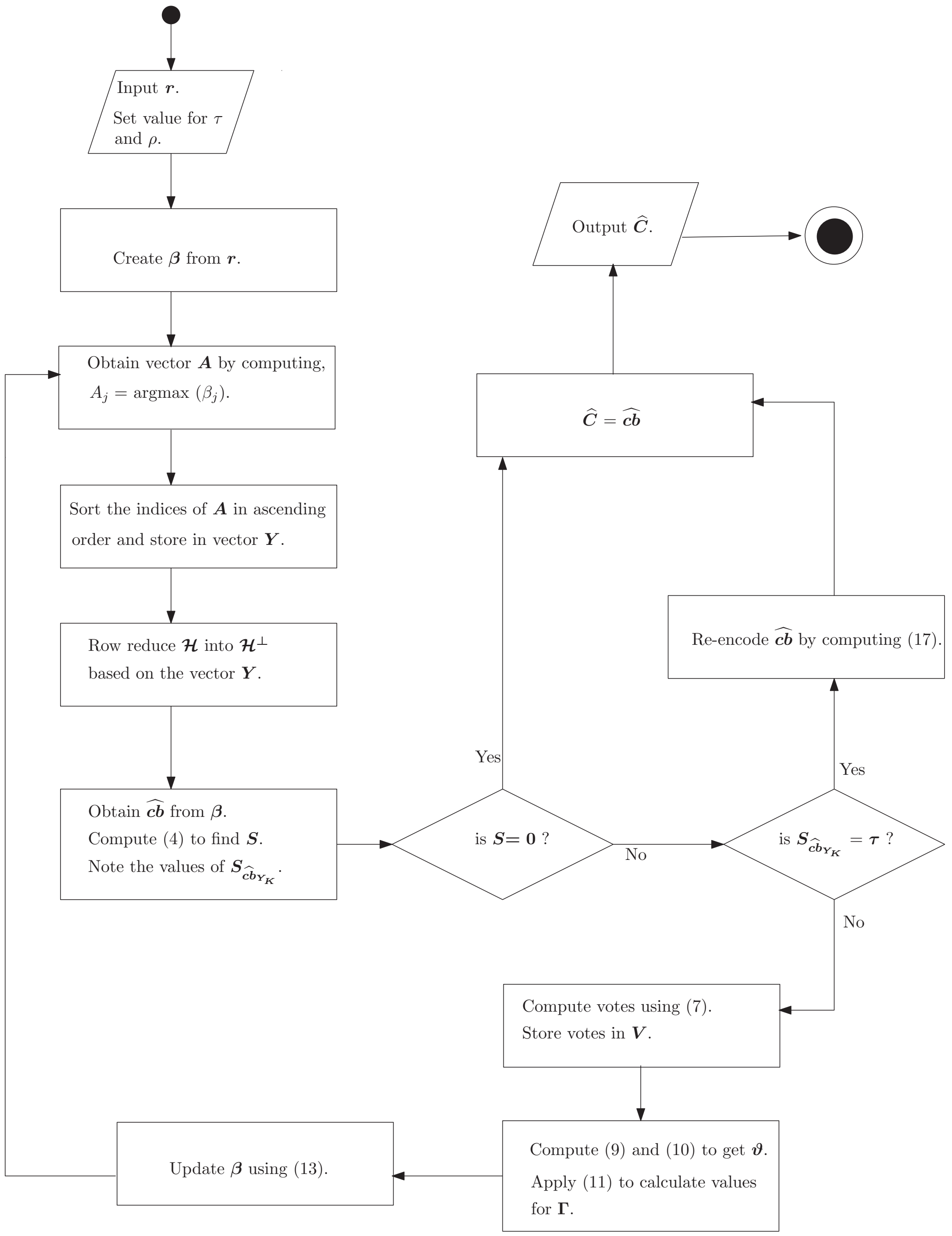

Fig. 1. Flow diagram representing the stages involved in the decoding process

transform version of the decoder and is denoted as $k\text{BD}_{nt}$ algorithm. As a result of the reduced Gaussian row reduction operations, the $k\text{BD}_{nt}$ algorithm has a lower computational complexity cost than the original implementation of the kBD algorithm. The same simulation parameters for encoding, modulation and transmission used to obtain the results in Fig. 2, are utilised for this set of simulations. The results for these simulations are presented in Fig. 6 and Fig. 7.

It can be seen from Fig. 6 that the kBD algorithm experiences a gain of 0.5dB when compared to the ABP algorithm with $\alpha = 0.05$ at a BER of 10^{-3}. The kBD algorithm also outperforms PTA_{bl} decoder, but with a smaller gain of about 0.4dB for the BER value of BER of 10^{-4}.

The $k\text{BD}_{nt}$ algorithm, being less complex due to its lack of iterative $\boldsymbol{\mathcal{H}}^{\perp}$ transformations, yields a similar performance to that of the ABP algorithm with $\alpha = 0.05$. However, the $k\text{BD}_{nt}$ algorithm is outperformed by both the kBD algorithm and the PTA_{bl} by about 0.65dB and 0.55dB respectively for a BER of 10^{-3}. It is important to note that the PTA_{bl} has a significantly higher computationally complexity cost when compared to all the bit-level decoders used in the simulation. This is because it performs Gaussian row reduction operations to transform the matrix $\boldsymbol{\mathcal{H}}$ for each of the iterations required during the decoding process, as seen in Fig. 7. This provides justification

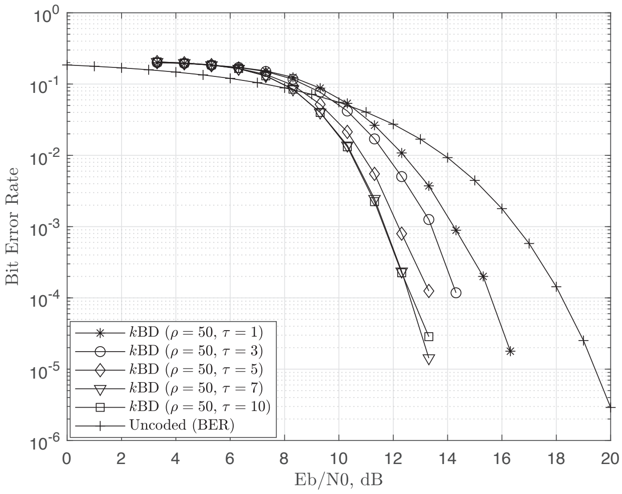

Fig. 2. Performance comparison of the kBD algorithm based on different values of τ in terms of BER for a (15,7) RS code.

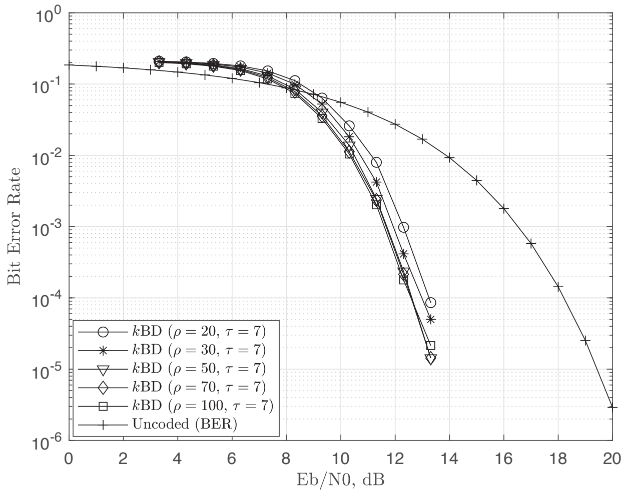

Fig. 4. Performance comparison of the kBD algorithm based on different values of ρ in terms of BER for a $(15, 7)$ RS code.

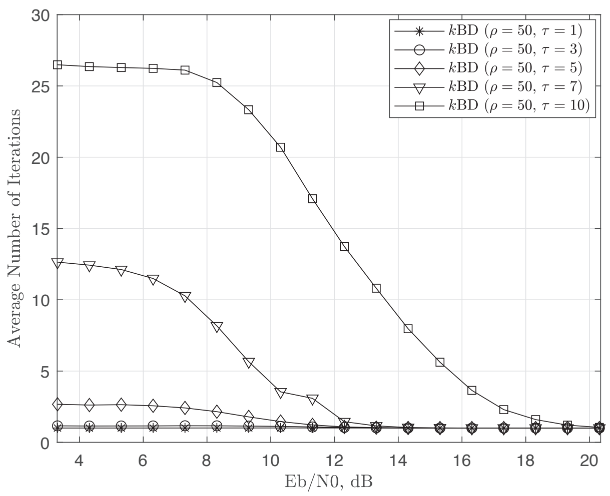

Fig. 3. Performance comparison of the kBD algorithm based on different values of τ in terms of average number of iterations for a (15,7) RS code.

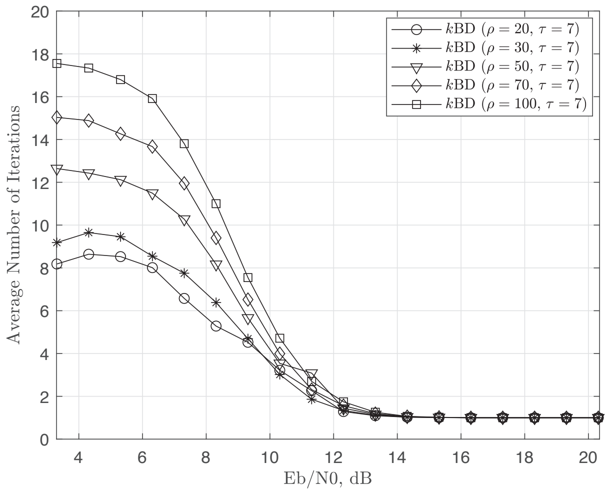

Fig. 5. Performance comparison of the kBD algorithm based on different values of ρ in terms of average number of iterations for a $(15, 7)$ RS code.

for the use of the $k\text{BD}_{nt}$ algorithm over the PTA_{bl} whenever a tradeoff is required between the algorithm complexity and the decoding performance.

From Fig. 6 and Fig. 7, the kBD algorithm is seen to be a better performing iterative bit-level soft-decision decoding algorithm. This is because it is able to achieve gains in BER performance, while running for fewer iterations than all other bit-level soft-decision decoding algorithms used in the $(15, 7)$ RS code simulations.

High rate codes

Additional simulations are carried out to test the performance of the proposed variations of the kBD algorithms under high rate conditions. Working at a high rate means working with a $\boldsymbol{\mathcal{H}}$ matrix with less rows when compared to a half rate code. That is, there are fewer syndrome check equations than in the case for the $(15, 7)$ RS code. This means that each of the K bits in the information set participate in less syndrome check equations.

With respect to this, a new set of simulations to determine an optimum value for τ are carried out. For these simulation, a $(15, 11)$ RS code is once again transmitted through an AWGN channel using a 16-QAM modulation scheme with Gray mapping. The results for these simulations are displayed in Fig. 8 and Fig. 9. It can be seen from Fig. 8 that the BER performance of the algorithm is the same for values of $\tau \geq 3$. The main difference in the performance of the algorithm can be seen in Fig. 9. The kBD algorithm with $\tau = 3$ requires less than half of the average number of iterations to decode the received vector, when compared to the kBD algorithm with values of $\tau \geq 5$. As mentioned in section III-A, the reason why the kBD algorithm with $\tau = 3$ requires less iterations than values of $\tau > 3$, is because the algorithm only has to ensure that each bit in the information set, $\widehat{cb}_{Y_K}$, satisfies 3 syndrome

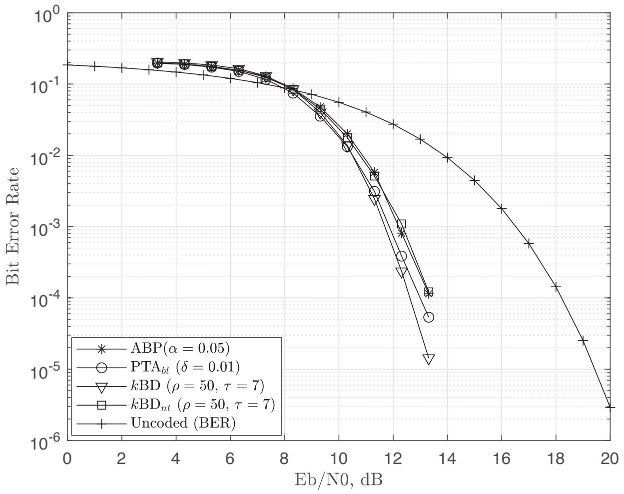

Fig. 6. Performance comparisons for bit-level decoders applied to a $(15, 7)$ RS code in terms of BER.

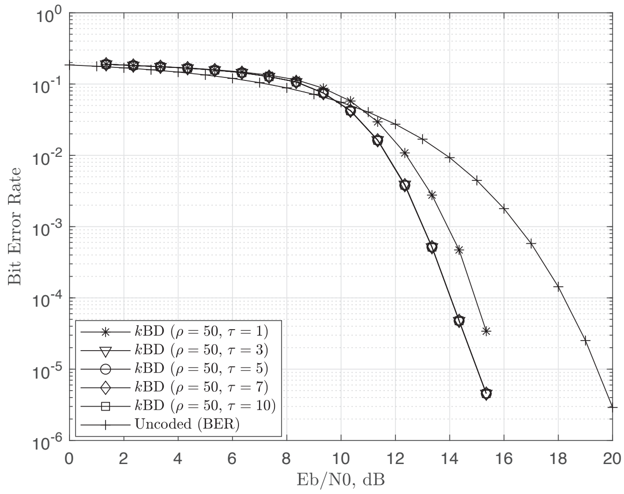

Fig. 8. Performance comparison of the kBD algorithm based on different values of τ in terms of BER for a $(15, 11)$ RS code.

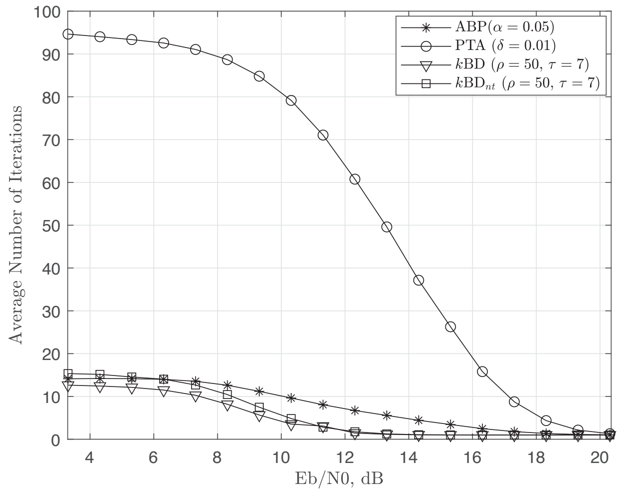

Fig. 7. Performance comparisons for bit-level decoders applied to a $(15, 7)$ RS code in terms of average number of iterations.

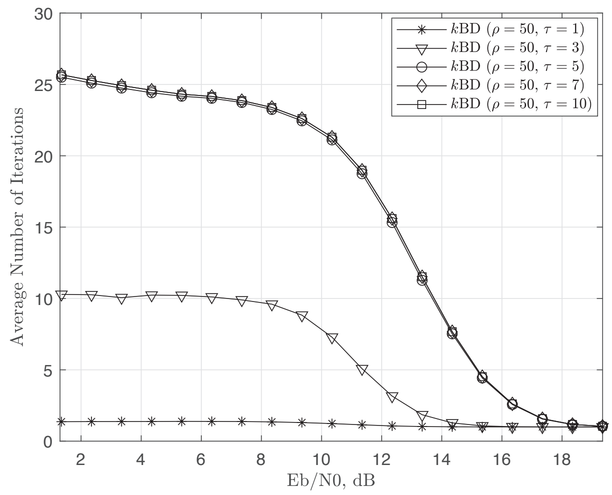

Fig. 9. Performance comparison of the kBD algorithm based on different values of τ in terms of average number of iterations for a $(15, 11)$ RS code.

check equations as opposed to the cases when $\tau > 3$. This enables the kBD algorithm to meet the decoding condition with fewer iterations than when $\tau \geq 5$. Also, the value of $\tau = 3$ is sufficient to give an optimum BER performance as a direct result of the $(15, 11)$ RS code having a $\boldsymbol{\mathcal{H}}$ matrix with less syndrome check equations. Hence, the kBD algorithm with $\rho = 50$ and $\tau = 3$ is used when benchmarking the performance of the decoder with the ABP and the PTA$_{bl}$ for high rate codes.

No modifications are made to the implementations of the PTA$_{bl}$ and the ABP algorithm from the case of the nearly half rate code. All the algorithms are run under the same conditions as the case for the (15,7) RS code. The results for this set of simulations can be seen in Fig. 10 and Fig. 11. From the results presented in Fig. 10, the performance of the kBD algorithm matches the performance of the PTA$_{bl}$. This performance is achieved by the kBD algorithm while running at a lower average number of iterations than the PTA$_{bl}$, during the decoding process, as seen in Fig. 11. The kBD algorithm also compares favourably to the ABP algorithm by yielding a slight BER performance gain of about 0.23dB at an BER value of 10^{-5}.

The kBD$_{nt}$ algorithm is only slightly outperformed by less than 0.1dB at a BER of 10^{-5} when compared to the PTA$_{bl}$ and achieves a similar BER performance to that of the ABP algorithm, while running at an average number iterations that is less than both algorithms. This justifies the use of the kBD$_{nt}$ algorithm, when selecting a bit-level soft-input soft-output decoder, in the case of a tradeoff between the algorithm complexity and the decoding performance.

The algorithm is also run for a $(31, 25)$ RS code so as to benchmark the proposed decoder against the results presented in [5]. For these simulations, the proposed algorithm is also tested against one of the modifications of the ABP decoder

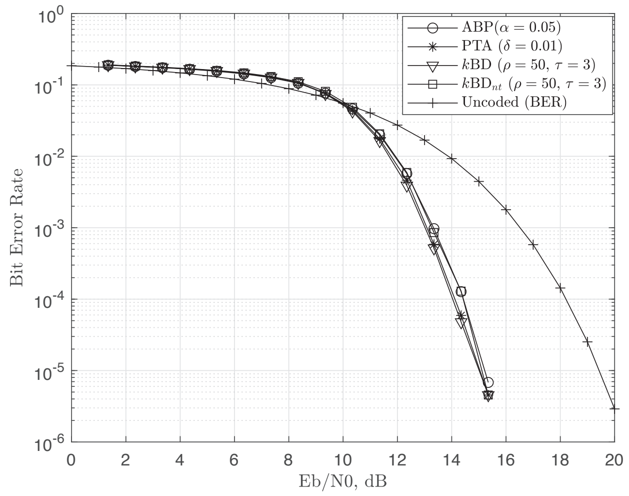

Fig. 10. Performance comparisons for bit-level decoders applied to a $(15, 11)$ RS code in terms of BER.

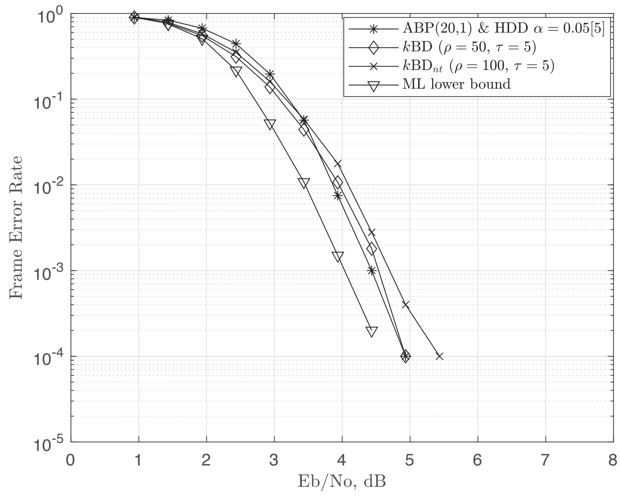

Fig. 12. Performance comparison of the ABP, PTA$_{bl}$, kBD algorithm and kBD$_{nt}$ algorithm for a $(31, 25)$ RS code.

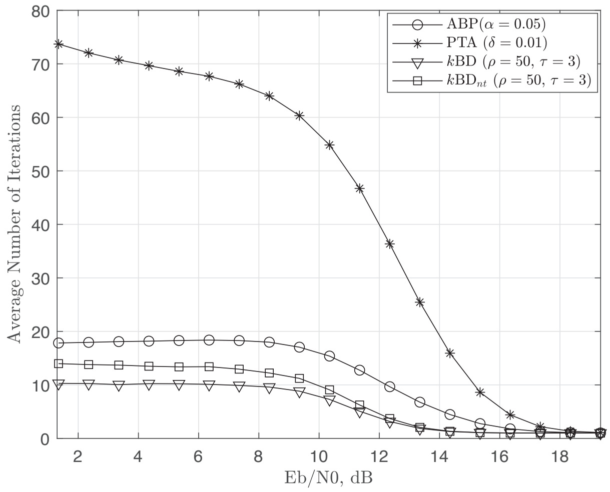

Fig. 11. Performance comparisons for bit-level decoders applied to a $(15, 11)$ RS code in terms of average number of iterations.

referred to as the ABP-HDD(20,1) algorithm. This version of the ABP works alongside a 'genie aided' hard decision decoder (HDD) [5]. This version of the ABP works iteratively, however, it does not converge to a codeword. Instead it runs for all the 20 iterations while outputting a codeword that is fed into the HDD with a genie aided stopping condition. The ABP only selects the most likely codeword if the HDD is not able to obtain the decoded vector from the list of codewords generated from each of the 20 iterations. The result of the ABP-HDD(20,1) is described as 'optimistic' in [5] due to the inclusion of the genie aided HDD. This is because the genie aided stopping criteria already knows the correct codeword and breaks the decoding process once the correct codeword is obtained instead of letting the iterative algorithm converge to the most likely codeword[5]. The genie aided HDD is added to the algorithm to prevent the decoder from running all 20 iterations and therefore speeding up the decoding process. This is because the algorithm is a double decoder. This means there is an increased complexity when compared to the generic version of the ABP. This is especially due to the algorithm running all 20 iterations during each decoding process, even for higher SNR values when the decoder requires less iterations to converge to a valid codeword.

The simulations are tested using BPSK modulation and the RS code is transmitted through an AWGN channel so as to obtain the results in a similar way to [5]. The Maximum Likelihood (ML) lower bound is obtained using the technique presented in [35] [42]. The results for these simulations are presented in Fig. 12. From the results in Fig. 12 it can be seen that the error correction performance of the kBD algorithm and kBD$_{nt}$ algorithm are quite favourable when compared to the generic ABP decoder. The kBD algorithm outperforms the generic version of the ABP by about 0.5dB at an FER of 10^{-3}. The less complex kBD$_{nt}$ algorithm yields a gain of about 0.25dB when compared to the generic form of the algorithm. However, the kBD algorithm has a loss of about 0.6dB when compared to the ML lower bound graph and a loss of less than 0.1dB when compared to the ABP-HDD(20,1) [5], for the same FER value.

V. Complexity Analysis

A. Time Complexity

In this section, the computational complexity cost for the kBD algorithm is compared to that of the ABP and the PTA$_{bl}$ decoders. The complexity cost of the bit-level soft-input soft-output decoders is measured in terms of the total number of operations carried out for the average number of iterations required in the entire decoding process. In order to better represent the computation calculations, notation is established. The average row weight and column weight of the participating bits indexed by t are denoted as W_r and W_c respectively. The average row weight of indices that match the parity submatrix of $\boldsymbol{\mathcal{H}}^{\perp}$ are denoted using W_k. The notation W_k is also used to denote the average row weight for computations that obtain the extrinsic bit information. The computational complexity cost for a single iteration of the ABP, the PTA$_{bl}$, kBD algorithm and the kBD$_{nt}$ algorithm are all summarised in the Table I, Table II, Table III and Table IV respectively.

For clarity, the complexity analysis of the ABP algorithm in Table I follows the decoding process described in [5]. The complexity analysis for the PTA$_{bl}$ in Table II follows the decoding process described in [4].

TABLE I
SUMMARY OF THE OVERALL COMPLEXITY FOR THE ABP DECODER.

Decoding Stage	Stage description	Number of Operations
Obtaining the vector $\|L\|$	Finding absolute values for L	N
Sorting of $\|L\|$	$\|L\|$ sorted in ascending order	N
Adapting the $\mathcal{H}$ matrix	Row reduction of $\mathcal{H}$	$M^2 \times N$
Obtaining the vector L_e	Calculating the extrinsic information	$(M \times W_k^2)+(N \times W_c)$ [6]
Updating the vector L	Adding $L_e(cb_j)$ to $L(cb_j)$	N
Obtaining the hard-decision vector $\widehat{cb}$	Assigning one of the binary values to each $L(cb_j)$	N
Syndrome Check	Performing the calculation represented in (9)	$M \times W_r$

Overall Complexity of the ABP decoder

Total $= N+N+(M^2 \times N)+(M \times W_k^2)+(N \times W_c)+N+N+(M \times W_r)$

Time complexity = $\mathcal{O}(M \times W_k^2)$ [6]

TABLE II
SUMMARY OF THE OVERALL COMPLEXITY FOR PTA$_{bl}$.

Decoding Stage	Stage description	Number of Operations
Finding the reliability vector	Searching for maximum values in β	N
Sorting of reliabilities	Reliabilities sorted in ascending order	N
Transforming matrix $\mathcal{H}$	Row reduction of $\mathcal{H}$	$M^2 \times N$
Obtaining the hard-decision vector $\widehat{cb}$	Assigning one of the binary values in β	N
Syndrome Check	Performing the calculation represented in (9)	$M \times W_r$
Correction step	Updating the β reliabilities based on each Syndrome check	$M \times W_r$
Scaling reliabilities in β	If $S \neq 0$, reliabilities are scaled such that they add up to 1 in preparation for the next iteration	$(N+2N)$

Overall Complexity of the PTA$_{bl}$

Total $= N+N+(M^2 \times N)+N+(M \times W_r)+\underbrace{(M \times W_r)+(N+2N)}_{\text{if } S \neq 0}$

Time complexity = $\mathcal{O}(M^2 \times N)$

TABLE III
SUMMARY OF THE OVERALL COMPLEXITY FOR THE kBD ALGORITHM.

Decoding Stage	Stage description	Number of Operations
Finding the vector $\beta_{\max}$	Searching for maximum values in β	N
Sorting the reliabilities	Reliabilities sorted in ascending order	N
Transforming matrix $\mathcal{H}$	Row reduction of $\mathcal{H}$	$M^2 \times N$
Obtaining the hard-decision vector $\widehat{cb}$	Assigning one of the binary values in β	N
Syndrome Check and vote tallying	Applying (9) and (12)	$(M \times W_r) + (M \times W_k^2)$
Obtaining the bit confidence rating	Applying (15),(14) and (16)	$(N+2N+2N)$
Updating β	Bit reliabilities are updated in β	$M \times W_r$
Checking for decoding condition	Checking if τ is met by the informational set	K
Scaling reliabilities in β	If $S \neq 0$ or τ is not met, reliabilities are scaled such that they add up to 1 in preparation for the next iteration	$(N+2N)$
Applying the Decoding condition	Applying (22)	$M \times W_k$ - only applied in the final iteration if $S \neq 0$ when the threshold τ is met

Overall Complexity of the kBD algorithm

Total $= N+N+(M^2 \times N)+N+(M \times W_r)+(M \times W_k^2)+(N+2N+2N)+(M \times W_r)+K+\underbrace{(N+2N)}_{\text{if } S \neq 0 \text{ or } \tau \text{ is not met}}+\underbrace{(M \times W_k)}_{\text{last iteration only}}$

Time complexity = $\mathcal{O}(M \times W_k^2)$

TABLE IV
SUMMARY OF THE OVERALL COMPLEXITY OF THE kBD$_{nt}$ ALGORITHM.

Decoding Stage	Stage description	Number of Operations
Finding the vector $\beta_{\max}$	Searching for maximum values in β	N
Sorting the reliabilities	Reliabilities sorted in ascending order	N
Transforming matrix $\mathcal{H}$	Row reduction of $\mathcal{H}$	$M^2 \times N$ - only applied in the first iteration.
Obtaining the hard-decision vector $\widehat{cb}$	Assigning one of the binary values in β	N
Syndrome Check and vote tallying	Applying (9) and (12)	$(M \times W_r) + (M \times W_k^2)$
Obtaining the bit confidence rating	Applying (15),(14) and (16)	$(N+2N+2N)$
Updating β	Bit reliabilities are updated in β	$(M \times W_r)$
Checking for decoding condition	Checking if the τ is met by informational set	K
Scaling reliabilities in β	If $S \neq 0$ or τ is not met, reliabilities are scaled such that they add up to 1 in preparation for the next iteration	$(N+2N)$
Applying the Decoding condition	If τ is met, (22) is computed	$(M \times W_k)$ - only applied in the final iteration if $S \neq 0$ when the threshold τ is met

Overall Complexity of the kBD$_{nt}$ algorithm

Total $= N+N+\overbrace{(M^2 \times N)}^{1^{st} \text{ iteration only}}+N+(M \times W_r)+(M \times W_k^2)+(N+2N+2N)+(N \times W_r)+K+\underbrace{(N+2N)}_{\text{if } S \neq 0 \text{ or } \tau \text{ is not met}}+\underbrace{(M \times W_k)}_{\text{last iteration only}}$

Time complexity = $\mathcal{O}(M \times W_k^2)$

To create a clear perspective of the complexities of the algorithms, additional tables that note the computations which involve ‘additions/subtraction’, ‘multiplications/divisions’ and ‘other’ operations are created for each algorithm. The tables only considers the operations that are unique to at least one algorithm. For example, operations like Gaussian elimination and the syndrome check are present in all algorithms and are therefore ignored in this analysis. The tables with this analysis are presented in Table V, Table VI, and Table VII respectively.

From Table. I, Table. III and Table. IV, it can be seen that the ABP decoder and both versions of the kBD algorithm all have the same time complexity. This is due both algorithms calculating the extrinsic information. However from computational complexities presented in Table. V and Table. VII, the extrinsic information computation for the ABP requires $(M \times W_k^2)$ multiplications as seen in [5] while both version of the kBD algorithm use $(M \times W_k^2)$ additions as shown in (12).

B. Complexity measured in Terms of Number of Operations

The research carried out also attempted to visually represent the complexities of the decoders. The complexity graphs plotted are based on the of total number of operations as a function of the average number of iterations run by the decoders for each SNR value. This additional set of complexity analysis simulations are carried out so as to investigate the effect of the iterative performance on the complexity cost of the algorithms. The total number operations are obtained from the equations given in Table I, Table II, Table III and Table IV and multiplied by the average number of iterations run by each algorithm for the different SNR values shown in Fig. 7 and Fig. 11.

TABLE V
SUMMARY OF THE COMPUTATIONAL COMPLEXITY FOR THE ABP DECODER.

Decoding Stage	$+/-$	$\times/\div$	other
Obtaining the vector $\|L\|$		N	
Obtaining the vector L_e	$(N \times W_c)$	$(M \times W_k^2)$	
Updating the vector L	N	N	

TABLE VI
SUMMARY OF THE COMPUTATIONAL COMPLEXITY FOR PTA$_{bl}$.

Decoding Stage	$+/-$	$\times/\div$	other
Finding the reliability vector			N
Correction step	$M \times W_r$		
Scaling reliabilities in β	N		$2N$

TABLE VII
SUMMARY OF THE COMPUTATIONAL COMPLEXITY FOR THE kBD ALGORITHM AND THE kBD$_{nt}$ ALGORITHM.

Decoding Stage	$+/-$	$\times/\div$	other
Finding the vector $\beta_{\max}$			N
vote tallying	$(M \times W_k^2)$		
Obtaining the bit confidence rating	N	$(2N+2N)$	
Updating β	$M \times W_r$		
Checking for decoding condition			K
Scaling reliabilities in β	N	$2N$	
Applying the Decoding condition	$M \times W_k$		

From the simulation performed for the (15,7) RS codes, it is found that $W_r = 14.74$, $W_c = 7.86$, $W_k = 13.74$, $M = 32$ and $N = 60$. The results using these values can be seen in Fig. 13.

It can be seen from Fig. 13 that both variants of the kBD algorithm require less operations than the ABP with $\alpha = 0.05$ and the PTA$_{bl}$ with $\delta = 0.01$. As expected, the kBD$_{nt}$ algorithm is the least complex of all the algorithms for the $(15, 7)$ RS code. It is important to note that the kBD$_{nt}$ algorithm is able to exhibiting a comparable BER performance to the high performance ABP algorithm, while being less complex, as shown in Fig. 6 and Fig. 13 for the $(15, 7)$ RS code. The kBD algorithm is also shown to yield a tolerable complexity when compared to both the ABP and the PTA$_{bl}$. For the low SNR values, the computational complexity cost of the kBD algorithm is comparable to the ABP. However, as the values of the SNR increase, the complexity cost of the kBD algorithm reduces significantly faster than the ABP decoder. From Table II, the PTA$_{bl}$ appears to be the least complex bit-level decoder for operations carried out in a single iteration. However, the numerous iterations required by the PTA$_{bl}$ to decode the received vector, considerably add to the computational complexity cost as seen from Fig. 13. This makes the PTA$_{bl}$ significantly more complex than the other bit-level decoders used in the simulations.

The low computational complexity cost of the kBD algorithm and the kBD$_{nt}$ algorithm, when compared to the ABP and the PTA$_{bl}$, is largely attributed to the iterative convergence rate of the decoder. The information set decoding based stopping criteria ensures the algorithm is able to converge to a codeword with less iterations. This is because the algorithm is only required to decode K bits. This is not the case for the ABP

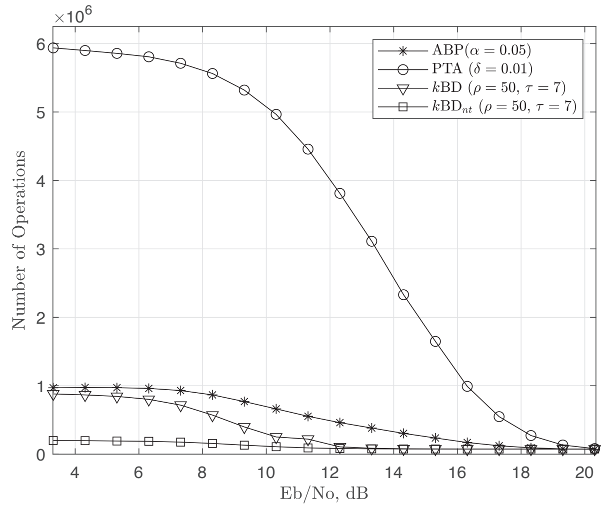

Fig. 13. Complexity comparisons for the bit-level decoders applied to a (15,7) RS code over an AWGN channel using a 16-QAM modulation scheme.

and the PTA_{bl} which have to decode the entire codeword of N bits before the iterative decoding process can break.
Simulations are also run for the high rate (15,11) RS code to compare the computational complexity cost for the bit-level decoders. The results for these simulations can be seen in Fig. 14. For these simulations $W_r = 22.48$, $W_c = 5.99$, $W_k = 21.48$, $M = 16$ and $N = 60$.
Similar to the results in Fig. 13, the variants of the kBD algorithm still perform less operations than the ABP and the PTA_{bl} during decoding of the received vector as seen in Fig. 14. However, there is a larger performance difference in the number of operations run between the kBD algorithm and the ABP for the $(15, 11)$ RS code, when compared to the $(15, 7)$ RS code. The main reason the kBD algorithm carries out less operations when compared to the ABP for the $(15, 11)$ RS code than the $(15, 7)$ RS code, is due to the use of a smaller value of τ. The smaller value of τ ensures the decoding condition is met much quicker. This reduces the number of iterations used to correct the bits in the received vector, which in turn reduces the number of operations required during the decoding process. Also, it is important to note that the ABP requires more iterations to decode the received vector for the $(15, 11)$ RS code when compared to the $(15, 7)$ RS code. This is because the high rate code has less rows and a larger parity submatrix in $\mathcal{H}$ when compared to the identity submatrix. This makes the matrix $\mathcal{H}$ more dense which affects the iterative convergence rate of the belief propagation algorithm. This is due to some of the unreliable bits saturating most of the checks which causes iterative decoding to be stuck at some pseudo-equilibrium points [5]. The results for the $k\text{BD}_{nt}$ algorithm are again quite favourable as in the case of the $(15, 7)$ RS code. This is because the significantly more complex PTA_{bl} only outperforms $k\text{BD}_{nt}$ algorithm by less than 0.1dB in terms of error correction. The $k\text{BD}_{nt}$ algorithm also matches the error correction performance of the ABP, which is also has a higher computational complex cost as seen in Fig. 10. As highlighted in section IV-B, this justifies the use of the algorithm whenever a tradeoff between the decoding performance and the algorithm complexity is required.

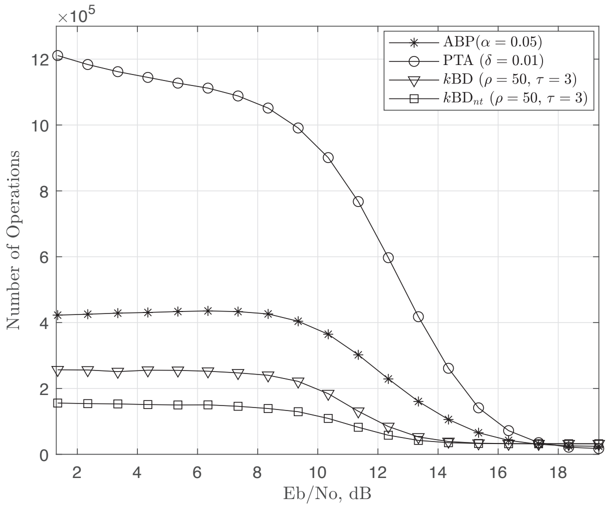

Fig. 14. Complexity comparisons for the bit-level decoders applied to a (15,11) RS code over an AWGN channel using a 16-QAM modulation scheme.

VI. Conclusion

In this paper an iterative soft-input soft-output bit-level decoder based on ISD is presented for RS codes. The algorithm has the advantage of working at a lower computational complexity cost while yielding a similar BER performance to the ABP and the bit-level PTA. The algorithm is able to perform at a lower complexity largely due to its low iterative convergence rate. The convergence rate of the proposed decoder is controlled by information set decoding techniques applied through an additional stopping criteria, referred to as the decoding condition, for the iterative decoding process. The decoding condition reduces the number of iterations required for decoding by enabling the decoder to output a decoded codeword of length N based on an information set of K bits. This approach reduces the iterative convergence rate because the algorithm only has to decode the information set made up of the most reliable bits.
The proposed decoder has two variants, the kBD algorithm and the less complex $k\text{BD}_{nt}$ algorithm. The kBD algorithm is able to match the error correction performance, and in some cases yield a slight gain in decoding performance, when compared to the ABP and the bit-level PTA. The $k\text{BD}_{nt}$ algorithm is only slightly outperformed, and in some case able match the performance, of the ABP and the bit-level PTA. However $k\text{BD}_{nt}$ algorithm is significantly less complex, due to the lack of row reduction operation being carried out iteratively, and can be used whenever a tradeoff is required between the algorithm complexity and the decoding performance.

VII. Future Recommendations

Work in this research focused on the development of a high performance decoding approach that runs at a relatively low complexity. The decoding approach works well, however improvements can still be made. Research can be carried out to test the coding gain the proposed algorithm can attain from using low weight parity check equations instead of the original ***H*** matrix as in the case for the parity check matrix extensions [43]. To further improve the error correction performance of the proposed bit-level decoder, at the expense of an increased computational complexity cost, double decoding techniques can also be explored. The use of a hard-decision decoder, in a similar way to the implementation with soft-decision decoders presented in [5], [16], [44], can also be investigated when applied to the proposed decoding approach.

References

[1] I. S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *J. Soc.Ind. Appl. Maths.*, 1960.
[2] R. Koetter and A. Vardy, "Algebraic Soft-Decision Decoding of Reed-Solomon Codes," *Information Theory, IEEE Transactions on*, 2003.
[3] O. O. Ogundile and D. J. J. Versfeld, "A Low Complexity Iterative Channel Estimation and Decoding Receiver Based on Reed-Solomon PTA," *IEEE Access*, vol. 4, pp. 8805–8813, 2016.

[4] O. Ogundile, Y. Genga, and D. Versfeld, “Symbol Level Iterative Soft Decision Decoder for Reed-Solomon Codes Based on Parity-Check Equations,” *Electronics Letters*, vol. 51, no. 17, pp. 1332–1333, Aug. 2015.

[5] J. Jiang and K. R. Narayanan, “Iterative Soft-Input Soft-Output Decoding of ReedSolomon Codes by Adapting the Parity-Check Matrix,” *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3746–3756, Aug 2006.

[6] L. Chen, “Iterative Soft Decoding of Reed-Solomon Convolutional Concatenated Codes,” *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4076–4085, October 2013.

[7] B. Kamali and A. Aghvami, “Belief Propagation Decoding of Reed-Solomon Codes; a Bit-Level Soft Decision Decoding Algorithm,” *Broadcasting, IEEE Transactions on*, vol. 51, no. 1, pp. 106–113, March 2005.

[8] V. T. Van, S. Mita, J. Li, C. Yuen, and Y. L. Guan, “Bit-Level Soft-Decision Decoding of Triple-Parity Reed-Solomon Codes Through Automorphism Groups,” *Communications Letters, IEEE*, vol. 17, no. 3, pp. 553–556, March 2013.

[9] Y. Genga, O. Oyerinde, and J. Versfeld, “Bit Level Implementation of the PTA Algorithm for Reed-Solomon Codes,” in *2017 Global Wireless Summit (GWS)*, Oct 2017, pp. 39–43.

[10] A. Vardy and Y. Be’ery, “Bit-Level Soft-Decision Decoding of Reed-Solomon Codes,” *Communications, IEEE Transactions on*, vol. 39, no. 3, pp. 440–444, Mar 1991.

[11] H. Xia and J. Cruz, “Performance of Reliability-Based Iterative Soft-Decision Reed-Solomon Decoding on Magnetic Recording Channels,” *IEEE Transactions on Magnetics*, vol. 43, no. 7, pp. 3320–3323, July 2007.

[12] Y. Yang, M. Jiang, and X. Wu, “An Investigation in Iterative Decoding of Reed-Solomon codes Based on Adaptive Belief Propagation,” in *2009 International Conference on Wireless Communications Signal Processing*, Nov 2009, pp. 1–5.

[13] J. Bellorado, A. Kavcic, M. Marrow, and L. Ping, “Low-Complexity Soft-Decoding Algorithms for ReedSolomon Codes-Part II: Soft-Input Soft-Output Iterative Decoding,” *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 960–967, March 2010.

[14] H. Lee, J. Wu, C. Wang, and Y. Ueng, “An Iterative Soft-Decision Decoding Algorithm for Reed-Solomon Codes,” in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 2775–2779.

[15] X. Huang and L. Chen, “Iterative Multistage Soft Decoding of Multilevel Reed-Solomon Codes,” in *2018 IEEE Information Theory Workshop (ITW)*, Nov 2018, pp. 1–5.

[16] M. El-Khamy and R. J. McEliece, “Iterative Algebraic Soft-Decision List Decoding of Reed-Solomon Codes,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 481–490, March 2006.

[17] D. J. C. MacKay, “Good Error-Correcting Codes Based on Very Sparse Matrices,” *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar 1999.

[18] F. MacWilliams and N. Sloane, *“The Theory of Error-Correcting Codes”*, 8th ed. Elsevier Science Publishers, June 1993, vol. 16, no. ISBN 0 444 85193 3, pp. 106.

[19] B. Liu, Y. Xie, L. Yang, and J. Yuan, “An Iterative Soft-Decision Decoding Algorithm with Dynamic Saturation for Short Reed-Solomon Codes,” in *2018 IEEE Information Theory Workshop (ITW)*, Nov 2018, pp. 1–5.

[20] J. Bellorado and A. Kavcic, “Low-Complexity Soft-Decoding Algorithms for Reed-Solomon Codes-Part I: An Algebraic Soft-In Hard-Out Chase Decoder,” *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 945–959, March 2010.

[21] D. Versfeld, J. Ridley, H. Ferreira, and A. Helberg, “On Systematic Generator Matrices for Reed-Solomon Codes,” *Information Theory, IEEE Transactions on*, vol. 56, no. 6, pp. 2549–2550, June 2010.

[22] Y. Genga, O. Ogundile, O. Oyerinde, and J. Versfeld, “A Low Complexity Encoder Construction for Systematic Quasi-Cyclic LDPC Codes,” in *2017 IEEE AFRICON*, Sept 2017, pp. 167–170.

[23] E. Prange, “The Use of Information Sets in Decoding Cyclic Codes,” *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, September 1962.

[24] J. T. Coffey and R. M. Goodman, “The Complexity of Information Set Decoding,” *IEEE Transactions on Information Theory*, vol. 36, no. 5, pp. 1031–1037, Sep 1990.

[25] M. Fossorier and S. Lin, “Reliability-Based Information Set Decoding of Binary Linear Codes,” in *Proceedings. 1998 IEEE International Symposium on Information Theory (Cat. No.98CH36252)*, Aug 1998, p. 229.

[26] L. Zhang, Z. Zhang, X. Wang, C. Zhong, and L. Ping, “Simplified Successive-Cancellation Decoding Using Information Set Reselection for Polar Codes with Arbitrary Blocklength,” *IET Communications*, vol. 9, no. 11, pp. 1380–1387, 2015.

[27] S. Scholl and N. Wehn, “Hardware Implementation of a Reed-Solomon Soft Decoder Based on Information Set Decoding,” in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014, pp. 1–6.

[28] Q. Guo, T. Johansson, E. Mrtensson, and P. Stankovski, “Information Set Decoding with Soft Information and Some Cryptographic Applications,” in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 1793–1797.

[29] C. Peters, “Information-Set Decoding for Linear Codes over F_q,” in *Sendrier N. (eds) Post-Quantum Cryptography. PQCrypto 2010. Lecture Notes in Computer Science*, vol. 6061, 2010, pp. 81–84.

[30] M. P. C. Fossorier and S. Lin, “Soft-Decision Decoding of Linear Block Codes Based on Ordered Statistics,” *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379–1396, Sep. 1995.

[31] I. Dumer, “Information-Set Soft-Decision Decoding,” in *1998 Information Theory Workshop (Cat. No.98EX131)*, Jun 1998, pp. 77–78.

[32] G. G. de Oliveira Brante, D. Nascimento Muniz, and W. Godoy, “Information Set Based Soft-Decoding Algorithm for Block Codes,” in *2010 IEEE Latin-American Conference on Communications*, Sep. 2010, pp. 1–6.

[33] A. Ahmed, R. Koetter, and N. R. Shanbhag, “Performance Analysis of the Adaptive Parity Check Matrix Based Soft-Decision Decoding Algorithm,” in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, vol. 2, Nov 2004, pp. 1995–1999 Vol.2.

[34] W. Sung and J. T. Coffey, “Information Set Decoding Complexity for Linear Codes in Bursty Channels with Side Information,” in *Proceedings of 1995 IEEE International Symposium on Information Theory*, Sep. 1995, p. 53.

[35] M. Bossert, *Channel Coding for Telecommuincations*. John Wiley & Sons, 1999, no. ISBN 0-471-98277-6.

[36] Y. Genga and D. Versfeld, “The Modified Soft Input Parity Check Transformation Algorithm for Reed Solomon Codes,” *SAIEE African Research Jornal*, vol. 108, no. 1, pp. 24–30, Mar. 2017.

[37] H. Xia and J. R. Cruz, “Reliability-Based Reed-Solomon Decoding for Magnetic Recording Channels,” *IEEE Transactions on Magnetics*, vol. 42, no. 10, pp. 2603–2605, Oct 2006.

[38] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Pearson Prentice Hall, 2004, no. ISBN 0-13-042672-5.

[39] T. K. Moon, *“Error Correction Coding: Mathematical Methods and Algorithms”*. John Wiley & Sons, 2005, no. ISBN 0-471-64800-0.

[40] S. Lin and W. Ryan, *“Channel Codes Classical and Modern”*. Cambridge University Press, 2009, no. ISBN-13 978-0-511-64182-4.

[41] F. Tosato and P. Bisaglia, “Simplified Soft-Output Demapper for Binary Interleaved COFDM with Application to HIPERLAN/2,” *HP Laboratories Bristol, Imaging Systems Laboratory*, no. HPL-2001-246, October 2001.

[42] R. Lucas and M. Bossert and M. Breitbach, “On Iterative Soft-Decision Decoding of Linear Binary Block Codes and Product Codes,” *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 276–296, Feb 1998.

[43] Y. Genga, O. Oyerinde, and J. Versfeld, “Improving the Decoding Performance of the PTA Algorithm for RS codes Through the Extension of the Parity Check Matrix,” in *2017 IEEE AFRICON*, Sept 2017, pp. 171–174.

[44] D. Chase, “Class of Algorithms for Decoding Block Codes with Channel Measurement Information,” *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 170–182, January 1972.

Yuval O. Genga received the B.Sc degrees in electrical and information engineering from the University of Nairobi, Nairobi, Kenya in 2012 and the M.Sc degree from the University of the Witwatersrand, Johannesburg, South Africa in 2016. He is a Ph.D. graduate from the University of the Witwatersrand, Johannesburg, South Africa. He is currently a sessional lecturer at the University of the Witwatersrand. His research interests include digital communications, digital transmission, forward error correction and signal processing techniques for wireless communication systems.

Olutayo O. Oyerinde (M'03–SM'18) received the Ph.D. degree in electronic engineering from the University of Kwazulu-Natal, Durban, South Africa, in 2011. He has been with the School of Electrical and Information Engineering, University of the Witwatersrand at Johannesburg, South Africa, since 2013, where he is currently an Associate Professor. He is an National Research Foundation (NRF) rated scientist, a Registered Professional Engineer (Pr.Eng.) with the Engineering Council of South Africa (ECSA), a Registered Engineer (R.Eng) with COREN, a Senior Member of Institute of Electrical and Electronics Engineer (SMIEEE), and a Corporate Member of NSE amongst others. He is an Associate Editor for the IEEE Access journal and also an editorial board member of the International Journal of Sensors, Wireless Communications & Control. His current research interests include wireless communications with specific interests in 5G and beyond 5G technologies, Cognitive Radio networks, and signal processing techniques for wireless communication systems.

Daniel J.J. Versfeld (M'01–SM'19) received the B.Eng degree in electronic engineering and the M.Eng degree in electronic engineering from North-West University, South Africa in 1999 and 2001, respectively, and the D.Ing (Electronic) degree from the University of Johannesburg, Johannesburg, South Africa, in 2011. From 2011 to 2016 he worked as a Senior lecture for the School of Electrical and Information Engineering, University of the Witwatersrand, South Africa. He is currently an associate professor at Stellenbosch University, Stellenbosch, South Africa. His research interests include signal processing applied to digital communications and forward error correction.

CONTENTS PAGE