

https://dspace.org/

# Symbol-level iterative information set decoding of RS codes

**Genga, Yuval; Oyerinde, Olutayo; Versfeld, Jaco**

**2023-07**
Elsevier
https://hdl.handle.net/10539/48806
Yuval Genga, Olutayo Oyerinde, Jaco Versfeld, Symbol-level iterative information set decoding of RS codes, Alexandria Engineering Journal, Volume 73, 2023, Pages 1-10, ISSN 1110-0168, https://doi.org/10.1016/j.aej.2023.04.034.

Downloaded from a DSpace repository. https://dspace.org/

Alexandria Engineering Journal (2023) 73, 1–10

HOSTED BY

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

# Symbol-level iterative information set decoding of RS codes

Yuval Genga [a,*], Olutayo Oyerinde [a], Jaco Versfeld [b]

[a] School of Electrical & Information Engineering, University of the Witwatersrand, Johannesburg, South Africa
[b] Department of Electrical and Electronic Engineering, Stellenbosch University, Matieland, South Africa

Received 1 December 2022; revised 28 March 2023; accepted 16 April 2023
Available online 27 April 2023

KEYWORD

Reed Solomon codes;
Soft-input soft-output decoding;
Iterative decoding;
Message passing decoding;
Information set decoding;
Decoding complexity

**Abstract** This paper presents the implementation of a low-complex iterative symbol-level decoding scheme for Reed-Solomon codes. Most soft-decision iterative decoders for Reed-Solomon codes work on a bit level due to the efficient passing of soft information due to the sparsity of the binary parity check matrix. These decoders yield a good error correction performance, but this comes at the cost of an increased computational complexity resulting from working at a bit-level. This study aims to lower the computational complexity of iterative decoding of Reed-Solomon codes by introducing a high-performance soft-decision iterative Reed-Solomon decoder that works on a symbol-level, in contrast to the bit-level implementation used in most iterative Reed-Solomon decoders. The proposed algorithm utilises soft information to decode while applying information set decoding techniques to an extended parity check matrix. The use of the extended parity check matrix provides additional parity check equations which assist the algorithm in determining the correct information set of symbols used in the decoding process. For a given $(n, k)$ Reed-Solomon code, the algorithm converges to a valid codeword by correctly decoding a specified information set of $k$ symbols from the received vector. Simulations run show the proposed algorithm performs favourably when compared to other symbol-level iterative decoders while working at a relatively lower complexity.

© 2023 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Engineering, Alexandria University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

## 1. Introduction

Forward error correction (FEC) is an error control strategy used in communication systems, deployed from the transmitter to the receiver. Error-correcting codes together with decoding algorithms are used to detect and correct errors that were introduced during transmission through a noisy communication channel. One of the most widely used error-correcting codes in FEC schemes are the Reed Solomon codes. Reed-Solomon (RS) codes are a class of non-binary Bose-Chaudhuri-Hocquenghem (BCH) codes that were developed in 1960 by Irving S. Reed and Gustave Solomon [1]. RS codes have played an important role in the telecommunications revolution with numerous applications in the Internet of Things (IoT), vehicular wireless communication technologies, digital storage devices, and satellite communication systems. Recently, RS encoding and transmission techniques have also been applied to optical communications [2,3].

* Corresponding author.
E-mail addresses: yuval.genga@wits.ac.za (Y. Genga), olutayo.oyerinde@wits.ac.za (O. Oyerinde), djjversfeld@sun.ac.za (J. Versfeld).

https://doi.org/10.1016/j.aej.2023.04.034
1110-0168 © 2023 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Engineering, Alexandria University.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

RS codes possess a convenient algebraic structure which enables the codes to detect and correct errors effectively, even in the presence of burst errors [4]. One of the key features of Reed-Solomon codes is their ability to correct errors that occur during transmission or storage. This is achieved by adding redundancy to the encoded data in the form of extra symbols called parity check symbols. These parity check symbols can be used to detect and correct errors that occur during transmission or storage. The popularity of RS codes has led to an active area of research focusing on improving their construction, developing efficient encoding techniques, and creating high-performance decoding schemes. [5–10,5].

Uncertainty methods are techniques that deal with the presence of noise, errors or ambiguity in data or models. They aim to quantify, propagate and estimate uncertainties in complex systems using probabilistic models, such as probability distributions or Bayesian inference. The most popular uncertainty method used in the iterative decoding of RS codes is the Adaptive Belief Propagation (ABP) algorithm [11]. This method is based on a soft-decision decoding approach that uses an iterative approach to update the likelihood of each possible codeword based on the received information. Uncertainty methods are not limited to FEC schemes and have applications in various fields [12–14].

The good error-correcting performance of the ABP when applied to RS codes is a direct result of the algorithm operating on a bit-level, which allows the decoder to make use of the sparse binary image of the matrix $\boldsymbol{H}$ [15,16]. As a result of this successful implementation, most iterative soft-decision RS decoders are in one way or another a variation of the ABP algorithm and, as a result, work on a bit-level [17–19,11,20]. These variants have been shown to output a favourable error-correcting performance when matched up to the original ABP decoder. However, this improved performance in error correction is realized at the cost of a higher computational complexity [17,11]. Decoding on a bit-level, which requires an $\boldsymbol{H}$ matrix that is comparatively larger than its symbol-level form, and applying Gaussian row reduction operations in each iteration, significantly adds to the increased cost in computational complexity of the ABP algorithm [11].

In an attempt to avoid the computational complexity cost of iteratively decoding soft information at a bit-level, research into the implementation of symbol-level decoders led to the development of the Parity-check Transformation Algorithm (PTA) [21]. The decoder works by reconstructing the matrix $\boldsymbol{H}$ based on the soft symbol reliability information of the RS code. The reliability information is then corrected based on the syndrome check equations in order to increase the likelihood of the correct symbol being selected. This process is repeated until the most likely decoded vector is obtained. The main novelty of the algorithm is that it can iteratively correct the soft information acquired from the received vector at the output of the channel based on the dense symbol-level $\boldsymbol{H}$ matrix. The decoding approach presented by the PTA is seen to exhibit better error-correcting performance when compared to the Koeter-Vardy (kV) [22] decoder and the Berlekamp Massey (BM) [23] decoding algorithm in [21,24]. However, it has to be noted that the decoder requires numerous iterations during the decoding process as a direct result of the dense composition of the $\boldsymbol{H}$ matrix [25]. The numerous iterations run by the PTA significantly contribute to its high cost in computational complexity.

Since it was first proposed by Prange [26], in all its forms, the Information Set Decoding (ISD) approach makes use of $k$ linearly independent symbols or bits obtained at the channel output to re-encode a distinct codeword [27]. The $k$ bits or symbols required for the re-encoding process are called the information set. Most implementations of the ISD approach used for error detection and correction in linear block codes have applications for binary codes. ISD algorithms mostly used in FEC also utilise list decoding techniques and Ordered Statistics Decoding (OSD) to correct the errors in the received vector [28–30]. One of the main advantages of using an ISD approach while decoding is that these algorithms have the benefit of a relatively lower complexity cost when compared to other decoding techniques [27] [31]. However, it has to be noted that ISD algorithms which utilise soft information are limited to soft-input hard-output decoders when implemented in conjunction with OSD and list decoding techniques. As a result, the implementation of these algorithms in circumstances where RS decoders are needed to fully utilise soft information derived from either the bits or symbols is restricted. This prohibits the use of ISD algorithms for applications that involve satellite transmission[32] or iterative decoding [33].

### *1.1. Motivation*

Reed-Solomon codes can be decoded either on a bit-level or a symbol-level. Bit-level decoding algorithms operate on the binary representation of the symbols. Most iterative soft-input soft-output decoders for RS codes operate on a bit-level due to the sparse nature of the binary $\boldsymbol{H}$ matrix. In contrast, the dense structure of the symbol-level $\boldsymbol{H}$ matrix hinders efficient message passing of soft information during the decoding process. Bit-level soft-decision iterative decoders offer favourable error correction performance. However, this approach has several limitations, such as increased complexity, and loss of information. Therefore, this research focuses on an iterative symbol-level decoding approach that directly estimates the transmitted symbols from the received signals without converting them to bits. The main aim of implementing an iterative symbol-level decoder for RS codes is to reduce the decoding computational complexity while potentially improving the decoding accuracy and preserving the information content of the symbols.

The PTA is a soft-decision symbol-level iterative decoder for RS codes that yields a favourable error correction performance when compared to other RS decoders. However, due to the dense nature of the $\boldsymbol{H}$ matrix, the PTA requires numerous iterations during the decoding process. Due to the challenges presented by having a dense structure in the symbol-level $\boldsymbol{H}$ matrix, investigations into the implementation of information set decoding using iterative soft-decision techniques at the symbol-level has yet to be fully explored.

This paper presents an error correction technique that allows for iterative decoding using a soft-input soft-output approach on the information set of the RS code implemented on a dense symbol-level $\boldsymbol{H}$ matrix. The proposed decoding technique adds extensions to the dense $\boldsymbol{H}$ matrix [34] intended to add sparsity while still working at a symbol-level. The extensions to the parity check matrix are made up of low-weight parity check equations that are added to the original symbol-level $\boldsymbol{H}$ matrix. The weight of all the parity check equations,

added to the $\boldsymbol{H}$ matrix, matches the minimum distance [4] of the code. Each extension to the parity check matrix has similar dimensions as the original $\boldsymbol{H}$ matrix. This enables the columns from each extension of the parity check matrix to be arranged in such a way that they generate different information sets that are unique to the corresponding extensions. The use of different information sets in each extension, coupled with the added sparsity of the extended $\boldsymbol{H}$ matrix, allows for a more efficient exchange of the soft information at a symbol level while decoding the received vector. This results in the reduced number of iterations required during the decoding process, which in turn yields a lower computational complexity cost for the proposed algorithm. Results from the simulations show that the proposed decoder yields a favourable performance with regard to error correction, iterative convergence and computational complexity when compared to the other soft-decision symbol-level iterative RS decoders.

## 2. Information set decoding principles and the importance of the parity check matrix extensions

Prior to presenting the proposed decoder implementation, the significance of the systematic structure in the parity check matrix $\boldsymbol{H}$, which is necessary for the information set decoding approach, is discussed as follows:

Consider an $(n,k)$ RS code $\mathscr{C}$ in the field GF($2^b$), where $n$ is the length of the codeword, $k$ is the length of the message and $b$ is a positive integer. The RS code has a systematic $\boldsymbol{H}$ matrix with the form.

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

where $\boldsymbol{I}$ and $\boldsymbol{Q}$ denote the identity submatrix and parity submatrix respectively. The $\boldsymbol{I}$ submatrix has the dimensions $(n-k)\times(n-k)$, while the submatrix $\boldsymbol{Q}$ has the dimensions $(n-k)\times k$. The RS codeword $\boldsymbol{c}$ is then transmitted through a noisy channel to get the received vector, $\boldsymbol{r}$. The soft information obtained from the vector $\boldsymbol{r}$ is then used to get the hard-decision vector $\widehat{\boldsymbol{c}}$, which is of the form given in (2).

$$\widehat{\boldsymbol{c}} = [\widehat{c}_1, \widehat{c}_2, \ldots, \widehat{c}_n]. \tag{2}$$

The vector $\widehat{\boldsymbol{c}}$ can be divided into two parts as shown in (3)

$$\begin{aligned} \boldsymbol{Y}_1 &= \left[\widehat{c}_1, \widehat{c}_2, \ldots, \widehat{c}_{(n-k)}\right], \\ \boldsymbol{Y}_2 &= \left[\widehat{c}_{(n-k)+1}, \ldots, \widehat{c}_n\right]. \end{aligned} \tag{3}$$

$\boldsymbol{Y}_1$ has a length of $(n-k)$ and matches the indices of the submatrix $\boldsymbol{I}$ and $\boldsymbol{Y}_2$ has a length of $k$ and matches the indices of submatrix $\boldsymbol{Q}$. Using the vectors $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$, the equation for the syndrome $\boldsymbol{S}$ can be expanded from 4–5 by computing the product between $\widehat{\boldsymbol{c}}$ and the transposed $\boldsymbol{H}$ matrix denoted as $\mathbf{H}^\top$

$$\boldsymbol{S} = \widehat{\boldsymbol{c}} \cdot \boldsymbol{H}^\top, \tag{4}$$

$$\boldsymbol{S} = \left(\boldsymbol{Y}_1 \cdot \boldsymbol{I}^\top\right) + \left(\boldsymbol{Y}_2 \cdot \boldsymbol{Q}^\top\right). \tag{5}$$

Due to the identity matrix submatrix $\boldsymbol{I}$, (5) can be reduced to

$$\boldsymbol{S} = \boldsymbol{Y}_1 + \left(\boldsymbol{Y}_2 \cdot \boldsymbol{Q}^\top\right). \tag{6}$$

Assuming $\widehat{\boldsymbol{c}}$ is a valid codeword, then the syndrome $\boldsymbol{S} = 0$. Therefore, the additive inverse property of the field GF$\left(2^b\right)$ [4] can be applied in (6) to give

$$Y_1 = Y_2 \cdot Q. \tag{7}$$

From (7) it can be seen that it is possible to decode the received vector of length $n$ with an information set of $k$ symbols, by utilising the systematic structure of $\boldsymbol{H}$. However, for the proposed decoding technique to successfully work, all the $k$ symbols in $Y_2$ have to be correct. This presents a challenge in the proposed decoding approach. This is because, for RS codes, the parity submatrix $\boldsymbol{Q}$ is completely dense. This means all the symbols in $Y_2$will participate in each syndrome check equation for each row of $\boldsymbol{H}$. This makes it difficult to detect and correct the symbols that are wrong in $Y_2$ in the iterative decoding process. To increase the chances of a successful implementation of the information set decoding approach on the dense $\boldsymbol{H}$ matrix, parity check matrix extensions are included in the decoding process [34]. This allows for the creation of multiple information sets depending on which matrix extension is being utilised in the iterative decoding process. This increases the chances of finding the most likely information set to use in the decoding of the remaining $(n-k)$ symbols.

## 3. The proposed iterative symbol-level decoding scheme

### 3.1. Error correction and detection

Similar to Section 2, Consider an $(n,k)$ RS codeword $c$ in GF $(2^b)$. The vector $r$ is obtained through the transmission of the codeword $c$ in a noisy channel. For transmission through a BPSK modulation scheme, the reliability matrix ($\boldsymbol{\beta}$) is constructed using the approach presented in [35,36]. Each of the noisy symbols of $r$ is made up of $b$ noisy bits as shown in (8).

$$r = ((r_{0,0}, \ldots r_{0,b-1}), (r_{1,0}, \ldots r_{1,b-1}), \ldots, (r_{n-1,0}, \ldots r_{n-1,b-1})). \tag{8}$$

For $0 \leqslant j \leqslant n$ and $1 \leqslant q \leqslant b$, the likelihood for each bit in $r$ is calculated as [36]

$$P^1_{r_{j,q}} = \frac{1}{\left(1 + exp\left(\left(4 \times u \times r_{(j,q)}\right)/N_0\right)\right)}, \tag{9}$$

where $N_0 = 1$ is the noise spectral density and $\pm u$ denotes the amplitude of the binary channel inputs. The notation $P^1_{r_{j,q}}$ represents the likelihood that the $q^{th}$ noisy bit of the $j^{th}$ symbols in the received vector is a 1. The likelihood that the $q^{th}$ noisy bit of the $j^{th}$ symbol in the received vector is a 0 is denoted as $P^0_{r_{j,q}}$ and is obtained using (10)

$$P^0_{r_{j,q}} = 1 - P^1_{r_{j,q}}. \tag{10}$$

The reliability matrix $\boldsymbol{\beta}$ is created using the bit likelihoods as presented in (11)

$$\beta_{(\alpha^z,n)} = \prod_{q=1}^{b} P^{\alpha^z_q}_{r_{j,q}}. \tag{11}$$

where $\alpha^z_q$ is the $q^{th}$ bit of the binary representation of the symbol $\alpha^z \in$ GF$\left(2^b\right)$. The notation $z = -\infty$ or $0 \leqslant z \leqslant 2^b - 2$. Please note that each element in the field GF($2^b$) is represented by a row in the $\boldsymbol{\beta}$ matrix as illustrated in [21]. The $\boldsymbol{\beta}$ matrix is then fed into the decoder. The decoding process for the proposed algorithm can be summarised as follows:

1. *Determining the symbol reliabilities*: In this stage, the highest reliabilities from each column in $\boldsymbol{\beta}$ are stored in vector $A$. This is expressed as

$$A = \left[A_1, A_2, \ldots, A_j\right] \tag{12}$$

where $A_j = \text{argmax}\left(\beta_j\right)$ for $0 \leqslant j \leqslant n$ and $\beta_j$ denotes the individual columns of $\boldsymbol{\beta}$. A sorting function is then applied to $A$ to ensure the reliabilities are presented in ascending order. The indexes for the largest $k$ reliabilities are then noted and placed in the vector $\mathscr{K}$. The vector $\mathscr{U}$ is used to store the remaining $(n-k)$ indexes which are considered to be unreliable. These vectors are shown in (13)

$$\begin{aligned} \mathscr{U} &= \left[l_1, l_2, \ldots, l_{(n-k)}\right], \\ \mathscr{K} &= \left[l_{(n-k)+1}, .., l_n\right]. \end{aligned} \tag{13}$$

where $l_j$ represents the symbol indexes.

2. *The* **H** *matrix extension*: The columns of matrix $H$ are then rearranged using row reduction techniques to ensure that columns with the indexes $\mathscr{K}$ match the parity submatrix. A computationally efficient way of doing this can be seen in [37,38]. The indexes used to rearrange the columns for the corresponding $H$ matrix extensions are obtained by using the equations shown in ()()()(14)–(16) [34].*for* $x = 1$

$$\begin{aligned} \mathscr{U}_{x=1} &= \left[l_1, l_2, \ldots, l_{(n-k)}\right], \\ \mathscr{K}_{x=2} &= \left[l_{(n-k)+1}, \ldots, l_n\right], \end{aligned} \tag{14}$$

*for* $x = 2$

$$\begin{aligned} \mathscr{U}_{x=2} &= \left[l_1, l_2, \ldots, l_{((n-k)-(x-1))}, l_{(n-k)+1}\right], \\ \mathscr{K}_{x=2} &= \left[l_{((n-k)-(x-2))}, l_{(n-k)+2}, \ldots, l_n\right], \end{aligned} \tag{15}$$

*for* $x \geqslant 3$

$$\begin{aligned} \mathscr{U}_{x\geqslant 3} &= \left[l_1, l_2, \ldots, l_{((n-k)-(x-1))}, l_{((n-k)-(x-1))+1}, \ldots, l_{(n-k)+1}\right], \\ \mathscr{K}_{x\geqslant 3} &= \left[l_{((n-k)-(x-2))}, l_{(n-k)+2}, \ldots, l_n\right], \end{aligned} \tag{16}$$

where $1 \leqslant x \leqslant \varepsilon$ and $\varepsilon$ represents the predefined number of parity check matrix extensions. The matrix extensions obtained from different index vectors are denoted as $h_x$. The complete extended parity check matrix is denoted as $H_\varepsilon$ and is expressed as

$$H_\varepsilon = \begin{bmatrix} h_1 \\ \vdots \\ h_\varepsilon \end{bmatrix}. \tag{17}$$

3. *The syndrome check*: The hard-decision vector $\widehat{c}$ is obtained from $\boldsymbol{\beta}$ and is used to get the syndrome, $S$, with the same approach shown in [21]. Each extension in $H_\varepsilon$ has a syndrome vector denoted as $s_x$. The syndrome equation for each of the vectors $s_x$ is calculated by getting the product between $\widehat{c}$ and the corresponding transposed matrix extensions in $H_\varepsilon$ as expressed in (18)

$$\begin{bmatrix} s_1 \\ \vdots \\ s_\varepsilon \end{bmatrix} = \widehat{c} \times \begin{bmatrix} h_1^\top \\ \vdots \\ h_\varepsilon^\top \end{bmatrix}. \tag{18}$$

4. *Symbol voting ratios*: Voting for each symbol is performed during the syndrome calculation. All symbols, excluding the symbol being looked into, participate in the vote. The syndrome equation presented in (18) can also be expressed in the form of the scalar product between the $\widehat{c}$ and each row of $H_\varepsilon$

$$s_{x_i} = \widehat{c} \cdot h_{x_i}, \tag{19}$$

where $1 \leqslant i \leqslant (n-k)$ represents the row in each matrix extension of $H$. Due to the rearranged systematic configuration of the parity check matrix extensions of $H_\varepsilon$, only one symbol index of $\mathscr{U}_x$ participates in each row check. Therefore, (19) can be reduced to the form shown in (20)

$$s_{x_i} = \widehat{c}_{t_i} \cdot h_{x_{(i,t_i)}}, \tag{20}$$

where $t_i$ has a length of $k+1$ and is used to denote the indices of the symbols in $\widehat{c}$ participating in the $i^{th}$ syndrome check equation of $s_x$. The indices denoted by the vector $t_i$ can be written as

$$t_i = \left\{n : h_{x_{(i,n)}} \neq 0\right\}. \tag{21}$$

To apply the vote, it is assumed all the symbols except the one being voted for are correct. It is also presumed that $s_{x_i} = 0$. This means the additive inverse property for the $\text{GF}\left(2^b\right)$ field [4] is applied to (20) to give

$$\widehat{c}_y \cdot h_{x_{(i,y)}} = \widehat{c}_{t\prime} \cdot h_{x_{(i,t\prime)}}, \tag{22}$$

where $y \in t_i$ denotes the index of the symbol that gets the vote. The subvector $t\prime$ represents all the indexes in the vector $t_i$ without the bit index $y$. To get the symbol vote for $\widehat{c}_y$, (22) is rewritten as

$$\widehat{c}_y = \frac{\widehat{c}_{t\prime} \cdot h_{x_{(i,t\prime)}}}{h_{x_{(i,y)}}}. \tag{23}$$

The actual value of $\widehat{c}_y$ obtained from (23) is an element in the field GF($2^b$) and is counted as a single vote for the respective element. This is done for all the participating symbols in all the rows of $H_\varepsilon$. The votes are tallied for each element in the field and placed in $V$ as shown

$$V = \begin{bmatrix} \alpha^{-\infty}, & \alpha^0, & \alpha^1 & \ldots & \alpha^{\left(2^b-2\right)} \\ V_{\alpha^{-\infty}}, & V_{\alpha^0}, & V_{\alpha^1} & \ldots & V_{\alpha^{\left(2^b-2\right)}} \end{bmatrix}. \tag{24}$$

The first row of $V$ represents each element in the field GF ($2^b$). The second row of the matrix $V$ has the values $V_{\alpha^z}$, where $z$ is either $-\infty$ or $0 \leqslant z \leqslant \left(2^b-2\right)$, which are used to represent the total number of votes each of the corresponding elements gets. The vote ratios are then calculated as seen in (25) and denoted using $\boldsymbol{\eta}$.

$$\boldsymbol{\eta} = \left[\frac{V_{\alpha^{-\infty}}}{V_T}, \frac{V_{\alpha^0}}{V_T}, \frac{V_{\alpha^1}}{V_T}, \ldots \frac{V_{\alpha^{\left(2^b-2\right)}}}{V_T}\right], \tag{25}$$

where $V_T$ represents the total number of votes cast and is expressed as

$$V_T = V_{\alpha^{-\infty}} + V_{\alpha^0} + \ldots + V_{\alpha^{\left(2^b-2\right)}}. \tag{26}$$

5. *Obtaining the confidence rating*: The algorithm uses the confidence ratios as a measure of assurance that a symbol is correct based on the number of votes it gets. The confidence rating of each symbol is stored in the vector $\mathbf{\Gamma}$ and is calculated as

$$\mathbf{\Gamma} = \frac{\boldsymbol{\eta}}{\rho}, \tag{27}$$

where $\rho$ represents a constant value defined in the initialisation stage of the decoding process. The confidence ratios for each symbol are represented as

$$\mathbf{\Gamma} = \left[\Gamma_{\alpha^{-\infty}}, \quad \Gamma_{\alpha^{0}}, \quad \ldots, \quad \Gamma_{\alpha^{\left(2^{b}-2\right)}}\right]. \tag{28}$$

6. *Updating* $\boldsymbol{\beta}$: The confidence ratios are used to update the reliabilities in the matrix $\boldsymbol{\beta}$. The update is based on the vote each symbol index gets in the voting stage. The update is performed by adding the confidence ratio of the symbol that wins the vote to its corresponding reliability in $\boldsymbol{\beta}$. This update is performed on each of the participating symbols for every row of $H_{\varepsilon}$ as shown.

$$\beta^{(f+1)}_{\left(\widehat{c}_{y,j}\right)} = \beta^{(f)}_{\left(\widehat{c}_{y,j}\right)} + \Gamma_{\widehat{c}_{y}}, \tag{29}$$

where $f$ represents the current iteration number and $\widehat{c}_y$ is used to represent the symbol obtained from (23) as indexed in the matrix $\boldsymbol{\beta}$. The matrix $\boldsymbol{\beta}$ is then scaled to ensure each column adds up to 1.

### 3.2. The stopping criteria

The steps for error detection and correction proposed in Section 3.1 are repeated until the stopping criteria are met. There are 2 stopping conditions defined for the proposed decoder. The main stopping condition is applied when the syndrome vector $S = 0$. The secondary stopping criteria is only applied when the algorithm meets the *decoding condition*. The decoding condition is defined as the moment when the algorithm determines that all the $k$ symbols matching the parity submatrix of any of the matrices $h_x$ are correct. Once the proposed symbol-level decoding algorithm deduces a correct information set of $k$ symbols, the entire codeword can be re-encoded using (7).

The proposed decoder determines if the $k$ information set symbols are correct by placing a threshold, $\tau$, on the total number of syndrome check equations where $s_x = 0$ on a single parity check matrix extension $h_x$. When the number of satisfied syndrome check equations matches the value of $\tau$ during the decoding process, the proposed algorithm assumes the decoding condition has been achieved. For example, if $\tau = 4$, then only 4 elements in a single syndrome vector $s_x$ should be equal to 0 for the decoding condition to be met. The remaining $(n-k)$ symbols matching the identity submatrix are then decoded using (7), based on the parity check matrix extension $h_x$ that matches the corresponding syndrome vector $s_x$.

Meeting the decoding condition enables the proposed decoder to run fewer iterations. This is because the decoding condition helps the algorithm to quickly determine if the $k$ information set symbols are correct as opposed to decoding the entire received codeword of length $n$. It has to be noted that the value of $\tau$ is used as a measure of confidence that the $k$ symbols matching the parity submatrix are correct. Lower values of $\tau$ ensure that the decoding condition is met early, which guarantees fewer iterations while decoding. However, this comes at the cost of a reduction in the error correction performance due to the increased chances of errors in the information set.

## 4. Results and analysis

### 4.1. Investigating the optimal conditions for the proposed symbol-level decoding algorithm

In this section, an analysis is carried out on the proposed decoder to determine its optimum conditions linked to the different number of parity check matrix extensions. The findings from the analysis are quantified based on the Symbol Error Rate (SER) to measure the error correction performance, and the average number of iterations to measure the decoding iterative convergence performance. For ease of reference, the proposed decoder is called the $k$-Symbol Decoder and is denoted as $k$SD. Similar to Section 3, the number of parity check matrix extensions used with the $k$SD algorithm are denoted as $\varepsilon$.

A close to half rate $(15, 7)$ RS code is used for the first set of tests. The variations of the $k$SD run in the simulations for this code include the decoder implemented with $\varepsilon = 2, \varepsilon = 3$ and $\varepsilon = 4$. All the variations of the $k$SD are run with $\rho = 20$ and $\tau = 3$. A BPSK modulation scheme is used to transmit the RS code through an AWGN channel. The simulation results are shown in Fig. 1 and Fig. 2.

The decoding performance exhibited by the $k$SD algorithm improves based on the number of extensions used, as seen in Fig. 1. The $k$SD with $\varepsilon = 4$ is the best-performing decoder out of the 3 algorithms. At an SER value of $10^{-4}$, the $k$SD with $\varepsilon = 4$ yields slight gains of about 0.05 dB and 0.15 dB when compared to the $k$SD with $\varepsilon = 3$ and $\varepsilon = 2$ respectively. The gains of the $k$SD algorithm with $\varepsilon = 4$ are more visible in terms of the average iterative convergence performance required for decoding the received vector as seen in Fig. 2. The $k$SD with $\varepsilon = 4$, at its worst iterative convergence performance, uses an average of just about 7.5 iterations to detect and correct errors

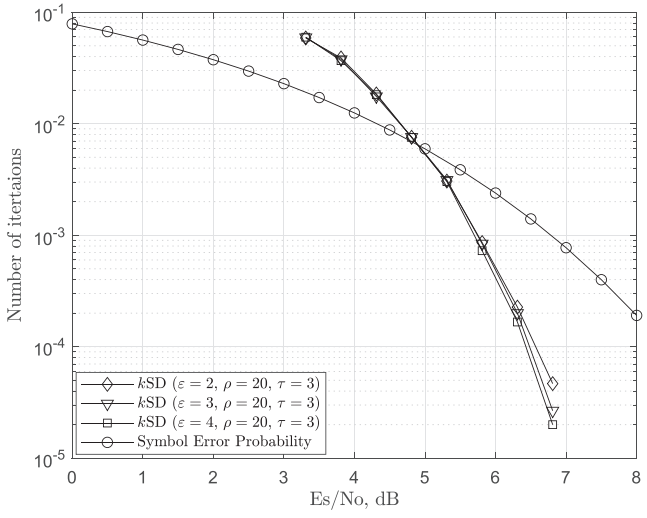


**Fig. 1** SER performance comparison of the $k$SD for a $(15, 7)$ RS code for different values of $\varepsilon$.

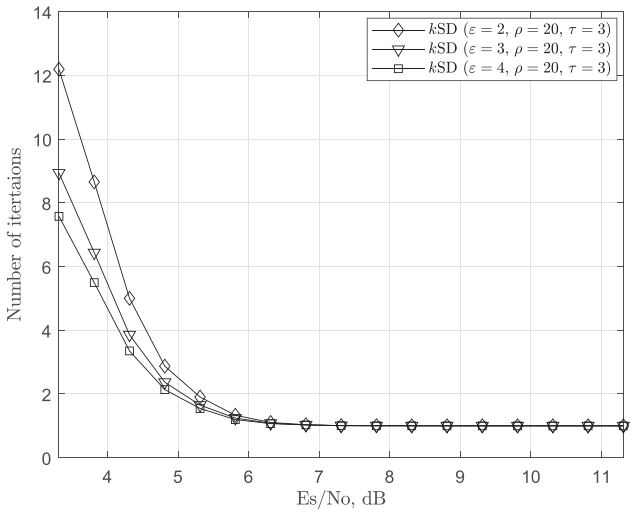


**Fig. 2** Iterative convergence performance comparison of the $k$SD for a $(15, 7)$ RS code for different values of $\varepsilon$.

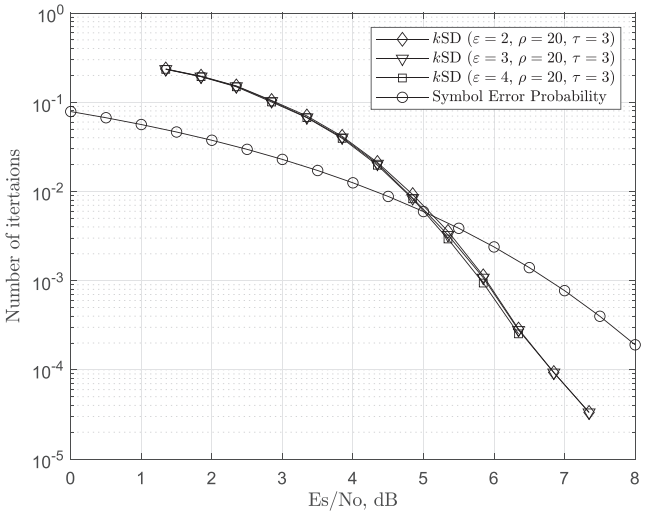


**Fig. 3** SER performance comparison of the $k$SD for a $(15, 11)$ RS code for different values of $\varepsilon$.

in the received vector. This is better than the $k$SD with $\varepsilon = 3$ and $\varepsilon = 2$ which, at their worst iterative convergence, run for an average of about 9 iterations and an average of just over 12 iterations respectively.

The proposed algorithm with $\varepsilon = 4$ exhibits relatively good performance in terms of SER and the average number of iterations because there are more combinations of symbol indexes of the vector $\mathscr{K}_x$ used with the different parity check matrix extensions $h_x$. This gives the algorithm more parity check equations which yield more syndrome vectors, $s_x$, that are used to check if the threshold is met. Using more syndrome vectors increases the chances of the decoding condition being met earlier, therefore decreasing the average number of iterations used in decoding the vector received at the output of the channel. It is also important to note that the larger size of $H_\varepsilon$ when $\varepsilon = 4$ enables the algorithm to make more updates to the $\beta$ matrix in a single iteration, which helps the algorithm converge to a codeword quicker. Due to the favourable iterative convergence performance exhibited by the $k$SD with $\varepsilon = 4, \rho = 20$ and $\tau = 3$, this version of the algorithm is used in the benchmarking tests for the nearly half rate case.

Similar tests are carried out on a high rate $(15, 11)$ RS code. Working with higher rates means implementing the proposed decoder on a $H$ matrix that has fewer rows, resulting in a smaller length of each vector $s_x$ obtained from the syndrome calculation in (18). Based on this, a smaller value of $\tau$ has to be used. For this set of simulations, the $k$SD is run with $\tau = 2$. The proposed algorithm is run under the same conditions with the same variants of the $k$SD algorithm as the nearly half rate case. The simulation results can be seen in Fig. 3 and Fig. 4.

Similar to the nearly half rate case, the proposed algorithm with $\varepsilon = 4$ is seen to yield a slightly better SER performance gain when matched up to the $k$SD with $\varepsilon = 3$ and $\varepsilon = 2$ as seen in Fig. 3. The $k$SD gives a coding gain of about 0.04 dB when compared to the proposed decoder with the $\varepsilon = 3$ and $\varepsilon = 2$ at an SER of $10^{-3}$. Similar to the nearly half rate case, the true gains of using the $k$SD with $\varepsilon = 4$ are seen in terms of the iterative convergence performance. The $k$SD with $\varepsilon = 4$, at its worst iterative convergence performance, is seen in Fig. 4 to run for an average number of just under 18 iterations. This

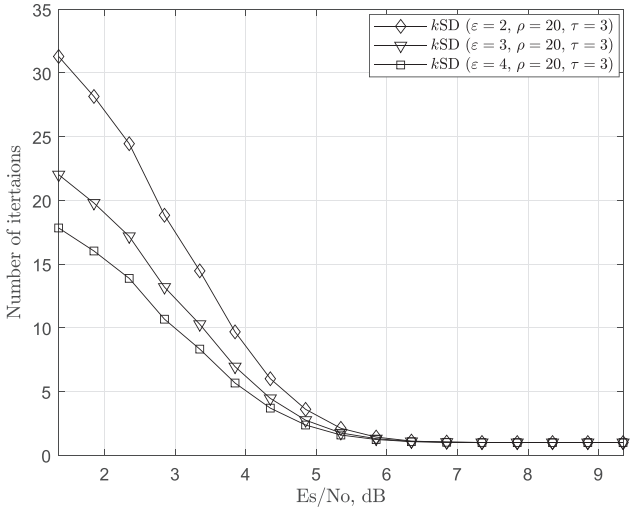


**Fig. 4** Iterative convergence performance comparison of the $k$SD for a $(15, 11)$ RS code for different values of $\varepsilon$.

iterative convergence performance is better than the $k$SD with $\varepsilon = 3$ and $\varepsilon = 2$, which run for an average number of just over 22 and just over 28 iterations at their worst iterative convergence respectively. Based on the iterative convergence results, the $k$SD with $\varepsilon = 4$ is used in the benchmarking simulations for the high rate codes.

### *4.2. Performance analysis of the kSD Algorithm*

In this section, an analysis into the performance of the proposed decoder is carried out when matched up against other iterative symbol-level decoders. The symbol-level implementations for the decoding algorithms are used in these simulations for the purpose of giving a fair comparison. The decoders used in the simulations are the symbol-level ABP algorithm and the PTA decoder.

#### 4.2.1. Half rate codes

For these simulations the symbol-level ABP decoder is run with the damping factor, $\theta$, having a value of 0.05 with the total number of iterations limited to a maximum of 20 at the initialisation of the algorithm [17]. While the PTA runs with a value of $\delta = 0.001$ [21]. The simulations are run under similar conditions as the previous tests for the $k$SD algorithm. The simulation results are presented in Fig. 5 and Fig. 6.

From Fig. 5, the $k$SD with $\varepsilon = 4$ is seen to be slightly outperformed by less than 0.05 dB when the error detection and correction performance is compared to that of the PTA decoder at an SER value of $10^{-3}$. The $k$SD algorithm, when matched up to the symbol-level ABP algorithm, can yield a gain of about 0.5 dB at an SER value of $10^{-4}$. From Fig. 6, the PTA is shown to run for numerous iterations while decoding the received vector, especially for the lower SNR regions. The PTA, at its worst iterative convergence performance, requires an average of about 643 iterations. This is a large number of iterations, especially when compared to the other iterative symbol-level decoders. The symbol-level ABP runs for just over 12 iterations. The $k$SD with $\varepsilon = 4$ can yield a better iterative convergence performance than all the algorithms, requiring an average maximum value of just about 7.5 iterations at its worst iterative convergence performance for the decoding process.

#### 4.2.2. High rate codes

A similar investigation on the performance of the $k$SD is carried out on a high rate $(15, 11)$ RS code. Simulation conditions used for the decoding algorithms remain the same as those considered for the close to half rate $(15, 7)$ RS code and are displayed in Fig. 7 and Fig. 8.

The $k$SD with $\varepsilon = 4$ is seen to match the error correction performance of the symbol-level ABP in Fig. 7. However, when compared to the PTA, the $k$SD with $\varepsilon = 4$ gives a gain of 0.3 dB at the SER value of $10^{-3}$. Similarly to the case of the $(15, 7)$ RS codes, the PTA algorithm requires the most average number of iterations to complete the error detection and correction process of the received vector. At its worst iterative convergence performance, the PTA runs for an average of about 775.8 iterations to complete the decoding process as seen in Fig. 8. The symbol-level ABP requires an average of about 19.2 iterations to converge to a codeword at its worst iterative convergence performance. The $k$SD with $\varepsilon = 4$ requires the least number of iterations for the detection and correction of errors in the received vector. At its worst iterative convergence performance, the $k$SD with $\varepsilon = 4$ uses an average of about 17.8 iterations to complete the decoding process.

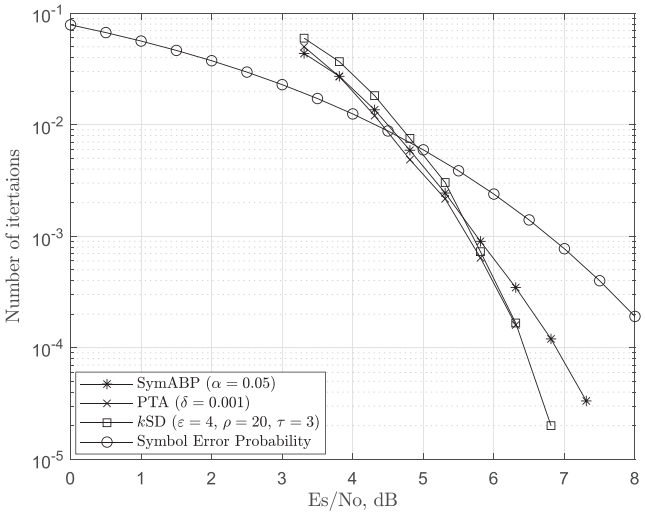


**Fig. 5** SER performance comparison of the Symbol level ABP, PTA and the $k$SD for a $(15, 7)$ RS code.

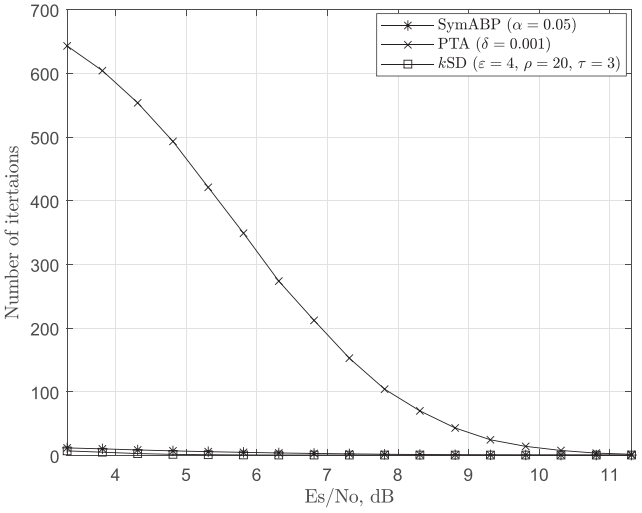


**Fig. 6** Iterative convergence performance comparison of the Symbol level ABP, PTA and the $k$SD for a $(15, 7)$ RS code.

## 5. Complexity analysis

In this section, an investigation is carried out to analyse the complexity cost of the proposed decoder. The analysis also compared the complexity cost to that of the PTA and the Symbol-level ABP. The complexity cost of the decoding algorithms is measured in terms of time complexity together with the number of additions/subtractions $(+/-)$ and multiplications/divisions $(\times/\div)$. The notation for the computational calculations are represented as follows; the total number of rows in $H$ are represented using the notation $m$. For the Symbol-level ABP, the average row weight of the participating symbols for the binary image of $H$ classified by $t$ are represented as $W_t$. The notation $W_c$ is used to symbolise the average column weight of the binary image of $H$. The dimensions of the binary image of $H$ are denoted as $mb \times nb$, where $mb = m \times b$ and $nb = n \times b$. The complexity of the decoding algorithms is summarised in Table 1.

From Table 1, the $k$SD is seen to have a complexity that is directly controlled by the number of parity check matrix extensions. The time complexity of the algorithm is also dominated by multiplications/divisions. However, the algorithm has much smaller dimensions when compared to the dimensions of the Symbol-level ABP which iteratively decodes the symbols based on the larger binary image of $H$. Also, the use of fewer iterations while decoding the received vector enables the $k$SD to operate at a relatively lower computational complexity. The PTA appears to be the least complex algorithm of all the three iterative decoders, however, the large number of iterations run

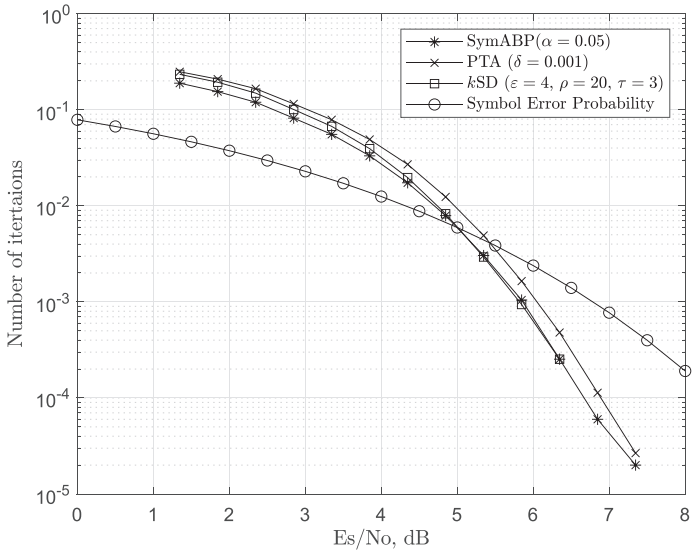


**Fig. 7** SER performance comparison of the Symbol level ABP, PTA and the $k$SD for a $(15, 11)$ RS code.

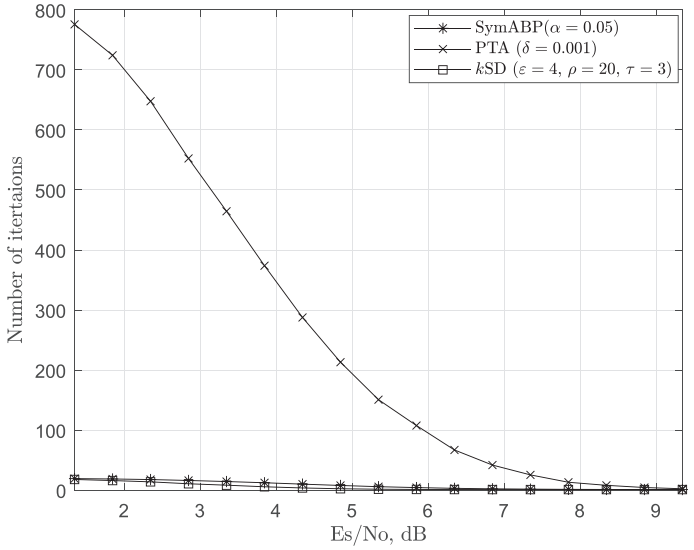


**Fig. 8** Iterative convergence performance comparison of the Symbol level ABP, PTA and the $k$SD for a $(15, 11)$ RS code.

**Table 1** Summary of the complexity analysis for the symbol-level iterative decoders.

| Decoder | Most complex stage | Time Complexity | $+/-$ | $\times/\div$ |
|---|---|---|---|---|
| SymABP | Obtaining extrinsic information | $\mathcal{O}(mb \times W_t^2)$ [17] | $(nb \times W_c)$ | $(mb \times W_t^2)$ |
| PTA | Transforming matrix $H$ | $\mathcal{O}(m^2 \times n)$ | $(m^2 \times n)$ [21] | $(m^2 \times n)$ |
| $k$SD | Calculating symbol votes | $\mathcal{O}((m \times \varepsilon) \times k^2)$ | | $((m \times \varepsilon)) \times (t + t^2)$ |

by the decoder while detecting and correcting errors in the received vector significantly adds to the overall cost in computational complexity.

## 6. Conclusion

A relatively low complexity iterative symbol-level soft-decision RS decoder is proposed in this paper. The algorithm has several variations depending on the number of parity check matrix extensions used. The extensions to the parity check matrix add to the number of parity check equations used in the decoding process of the $k$SD. Rearranging the columns of the different parity check matrix extensions in line with the symbol reliability information, help in identifying errors based on the syndrome vectors obtained from the parity check equations. Using the parity check matrix extensions and placing a threshold on the total number of satisfied syndrome

check equations by the $k$ symbols in the information set, improves the iterative convergence performance of the decoder. This in turn makes the decoder more efficient. Adding more parity check matrix extensions increases the complexity of the proposed decoder for a single iteration. However, the overall SER and iterative convergence performance is improved. This provides a justifiable tradeoff when selecting the number of parity check matrix extensions required during the decoding process.

The paper also shows the implementation of information set decoding techniques on a symbol-level. The proposed algorithm can converge to a valid codeword by correctly decoding a specified information set of k symbols from the received vector. This is especially significant for symbol-level iterative decoding schemes because of the dense nature of the symbol-level H matrix for RS code. The information set decoding implementation helps to significantly reduce the number of iterations required for decoding which in turn helps reduce the overall complexity of the decoder.

The proposed decoder performs favourably when compared with the existing symbol-level RS decoders. In comparison with the PTA decoder, the proposed algorithm with $\varepsilon = 4$ matches the performance for the close to half rate $(15, 7)$ RS code. This performance is carried over to higher rate codes as the same version of the decoder yields similar performance to the symbol-level ABP for the $(15, 11)$ RS code. However, it has to be noted that the error correction performance exhibited by the Symbol-level ABP and the PTA comes at the expense of the decoders operating at a higher computational complexity cost than the $k$SD with $\varepsilon = 4$. This gives justification for the use of the $k$SD in situations where a performance-complexity tradeoff is necessary.

For future works, several other potential techniques can be investigated in the selection of the index vectors used to rearrange the different parity check matrix extensions $h_x$. This would help in the selection of a more diverse set of vectors used in obtaining the parity check matrix extensions. This will help to improve the performance and efficiency of the proposed algorithm without a significant increase in complexity.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

The financial assistance of the National Research Foundation (NRF) of South Africa towards this research is hereby acknowledged. The financial support of the Centre for Telecommunication Access and Services (CeTAS), The University of the Witwatersrand, Johannesburg, South Africa is also acknowledged.

## References

[1] I.S. Reed, G. Solomon, Polynomial Codes over Certain Finite Fields, J. Soc. Ind. Appl. Maths.

[2] G. Karpagarajesh, R. Santhana Krishnan, Y. Harold Robinson, S. Vimal, S. Kadry, Y. Nam, Investigation of digital video broadcasting application employing the modulation formats like QAM and PSK using OWC, FSO, and LOS-FSO channels, Alexandria Eng. J. 61 (1) (2022) 647–657, https://doi.org/10.1016/j.aej.2021.06.038, URL: https://www.sciencedirect.com/science/article/pii/S1110016821004014.

[3] M. Singh, A. Atieh, M.H. Aly, S.A. El-Mottaleb, 20 Gbps SAC-OCDMA-OAM-based FSO transmission system: Performance evaluation under different weather conditions, Alexandria Eng. J. 61 (12) (2022) 10407–10418, https://doi.org/10.1016/j.aej.2022.03.070, URL: https://www.sciencedirect.com/science/article/pii/S1110016822002459.

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

[5] S. Tang, S. Cai, X. Ma, A new chase-type soft-decision decoding algorithm for Reed-Solomon codes, Alexandria Eng. J. 61 (12) (2022) 13067–13077, https://doi.org/10.1016/j.aej.2022.07.007, URL: https://www.sciencedirect.com/science/article/pii/S1110016822004604.

[6] A. Neri, Twisted linearized Reed-Solomon codes: A skew polynomial framework, J. Algebra 609 (2022) 792–839, https://doi.org/10.1016/j.jalgebra.2022.06.027, URL: https://www.sciencedirect.com/science/article/pii/S0021869322003295.

[7] Y.J. Tang, X. Zhang, Fast En/Decoding of Reed-Solomon Codes for Failure Recovery, IEEE Trans. Comput. 71 (3) (2022) 724–735, https://doi.org/10.1109/TC.2021.3060701.

[8] F. Ivanov, E. Krouk, V. Zyablov, New Code-Based Cryptosystem Based on Binary Image of Generalized Reed-Solomon Code, in, XVII International Symposium Problems of Redundancy in Information and Control Systems (REDUNDANCY) 2021 (2021) 66–69, https://doi.org/10.1109/REDUNDANCY52534.2021.9606467.

[9] J. Ye, T.-Y. Wu, J. Xing, L. Chen, Iterative Soft Decoding of Reed-Solomon Tail-Biting Convolutional Concatenated Codes, in, International Conference on Wireless Communications and Signal Processing (WCSP) 2020 (2020) 957–961, https://doi.org/10.1109/WCSP49889.2020.9299744.

[10] A.M. Veresova, A.A. Ovchinnikov, Comparison of the Probability of Reed? Solomon and LDPC Codes Decoding Error in the Gilbert? Elliott Channel, in, Wave Electronics and its Application in Information and Telecommunication Systems (WECONF) 2022 (2022) 1–4, https://doi.org/10.1109/WECONF55058.2022.9803501.

[11] Y. Yang, M. Jiang, X. Wu, An investigation in iterative decoding of reed-solomon codes based on adaptive belief propagation, Int. Conf. Wireless Commun. Signal Process. 2009 (2009) 1–5, https://doi.org/10.1109/WCSP.2009.5371418.

[12] A.R. Abbasi, Probabilistic load flow based on holomorphic embedding, kernel density estimator and saddle point approximation including correlated uncertainty variables, Electric Power Syst. Res. 183 (2) (2020), https://doi.org/10.1016/j.epsr.2019.106178, URL: https://www.sciencedirect.com/science/article/pii/S0378779619304973106178.

[13] M.R.A.A. Abdollah Kavousi-Fard, Reza Khorram-nia, An Smart Stochastic Approach to Model Plug-in Hybrid Electric Vehicles Charging Effect in the Optimal Operation of Micro-grids, J. Intell. Fuzzy Syst. 28 (2015) 835–842, https://doi.org/10.3233/IFS-141365.

[14] A.A.S.T. Abdollah Kavousi-Fard, Somayeh Abbasi, Optimal probabilistic reconfiguration of smart distribution grids considering penetration of plug-in hybrid electric vehicles, J. Intell. Fuzzy Syst. 29 (2015) 1847–1855, https://doi.org/10.3233/IFS-151663.

[15] M. El-Khamy, R.J. McEliece, Iterative algebraic soft-decision list decoding of Reed-Solomon Codes, IEEE J. Sel. Areas Commun. 24 (3) (2006) 481–490, https://doi.org/10.1109/JSAC.2005.862399.

[16] F. MacWilliams, N. Sloane, The Theory of Error-Correcting Codes, 8th Edition, Vol. 16, Elsevier Science Publishers, 1993.

[17] J. Jiang, K.R. Narayanan, Iterative soft-input soft-output decoding of reed?solomon codes by adapting the parity-check matrix, IEEE Trans. Inf. Theory 52 (8) (2006) 3746–3756, https://doi.org/10.1109/TIT.2006.878176.
[18] L. Chen, Iterative soft decoding of Reed-Solomon convolutional concatenated codes, IEEE Trans. Commun. 61 (10) (2013) 4076–4085, https://doi.org/10.1109/TCOMM.2013.082813.120943.
[19] H. Lee, J. Wu, C. Wang, Y. Ueng, An iterative soft-decision decoding algorithm for reed-solomon codes, IEEE Int. Symp. Inform. Theory (ISIT) 2017 (2017) 2775–2779, https://doi.org/10.1109/ISIT.2017.8007035.
[20] W. Zhang, S. Zou, Y. Liu, Iterative soft decoding of reed-solomon codes based on deep learning, IEEE Commun. Lett. 24 (9) (2020) 1991–1994, https://doi.org/10.1109/LCOMM.2020.2992488.
[21] O. Ogundile, Y. Genga, D. Versfeld, Symbol level iterative soft decision decoder for reed-solomon codes based on parity-check equations, Electron. Lett. 51 (17) (2015) 1332–1333, https://doi.org/10.1049/el.2015.1932.
[22] R. Koetter, A. Vardy, Algebraic Soft-Decision Decoding of Reed-Solomon Codes, Information Theory, IEEE Transactions on doi:10.1109/TIT.2003.819332.
[23] J. Massey, Shift-register synthesis and BCH Decoding, IEEE Trans. Inform. Theory 15 (1) (1969) 122–127, https://doi.org/10.1109/TIT.1969.1054260.
[24] O.O. Ogundile, D.J.J. Versfeld, A low complexity iterative channel estimation and decoding receiver based on Reed-Solomon PTA, IEEE Access 4 (2016) 8805–8813, https://doi.org/10.1109/ACCESS.2016.2633285.
[25] Y. Genga, D. Versfeld, The modified soft input parity check transformation algorithm for reed solomon codes, SAIEE African Res. J. 108 (1) (2017) 24–30, https://doi.org/10.23919/SAIEE.2017.8531513.
[26] E. Prange, The use of information sets in decoding cyclic codes, IRE Trans. Inform. Theory 8 (5) (1962) 5–9, https://doi.org/10.1109/TIT.1962.1057777.
[27] J.T. Coffey, R.M. Goodman, The complexity of information set decoding, IEEE Trans. Inf. Theory 36 (5) (1990) 1031–1037, https://doi.org/10.1109/18.57202.
[28] M. Fossorier, S. Lin, Reliability-Based Information Set Decoding of Binary Linear Codes, in: Proceedings. 1998 IEEE International Symposium on Information Theory (Cat. No.98CH36252), 1998, p. 229. doi:10.1109/ISIT.1998.708833.
[29] I. Dumer, Information-Set Soft-Decision Decoding, in: 1998 Information Theory Workshop (Cat. No.98EX131), 1998, pp. 77–78. doi:10.1109/ITW.1998.706443.
[30] S. Scholl, 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) 2014 (2014) 1–6, https://doi.org/10.7873/DATE.2014.222.
[31] W. Sung, 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, 1995, p. 53. doi:10.1109/ISIT.1995.531155.
[32] A. Vardy, Y. Be'ery, Bit-level soft-decision decoding of reed-solomon codes, IEEE Trans. Commun. 39 (3) (1991) 440–444, https://doi.org/10.1109/26.79287.
[33] M. Bossert, Channel Coding for Telecommuincations, no, John Wiley & Sons, 1999, ISBN 0-471-98277-6,.
[34] Y. Genga, O. Oyerinde, J. Versfeld, Improving the decoding performance of the PTA Algorithm for RS codes through the extension of the parity check matrix, IEEE AFRICON 2017 (2017) 171–174, https://doi.org/10.1109/AFRCON.2017.8095476.
[35] M. Davey, D. MacKay, Low density parity check codes over GF (q), in: 1998 Information Theory Workshop (Cat. No.98EX131), 1998, pp. 70–71. doi:10.1109/ITW.1998.706440.
[36] S. Lin, W. Ryan, Channel Codes Classical and Modern, no. ISBN-13 978-0-511-64182-4, Cambridge University Press, 2009.
[37] D. Versfeld, J. Ridley, H. Ferreira, A. Helberg, On systematic generator matrices for reed-solomon codes, IEEE Trans. Inform. Theory 56 (6) (2010) 2549–2550, https://doi.org/10.1109/TIT.2010.2046104.
[38] Y. Genga, O. Ogundile, O. Oyerinde, J. Versfeld, A low complexity encoder construction for systematic quasi-cyclic LDPC codes, IEEE AFRICON 2017 (2017) 167–170, https://doi.org/10.1109/AFRCON.2017.8095475.