Two Dimensional Algebraic Error Correcting Codes

Shounak Roy and Shayan Srinivasa Garani
Department of Electronic Systems Engineering
Indian Institute of Science, Bengaluru: 560012, India
Email: {shounak, shayangs}@iisc.ac.in

***Abstract*—Using a frequency domain approach, we construct two dimensional (2D) codes for correcting random error patterns. We provide algorithms to find the erroneous locations and compute the entire error spectrum. Our code construction is algebraic, and is useful for encoding 2D data for applications ranging from bar codes to imaging applications.**

***Index Terms*—2D error correcting codes, 2D finite field Fourier transform, cyclic codes, error locator ideals.**

I. Introduction

Over the past few years we have seen a paradigm change in data storage technologies. These technologies are shifting towards two-/multi- dimensional structures as they offer improved signal-to-noise (SNR) ratio and format efficiencies [1] which has lead to higher storage densities. Examples of such storage technologies include 2D magnetic recording [2], bit-patterned media [3], 3D flash storage [4], optical holographic recording [5] etc. The inherent data format of these technologies are in 2D, which is useful to combat 2D intersymbol-interference [6], noise and other impairments, leading to enhanced storage density. The data to be encoded and written onto the storage medium must be capable of correcting burst and random errors. This leads to the design of efficient of 2D error correcting codes along with modulation codes for these storage channels, along with sophisticated signal processing for timing recovery, equalization and detection [6], [7], [8]. 2D error correction can be iterative [9] or algebraic. Though iterative codes provide superior performance than algebraic codes, in the waterfall region, the latter is preferred in applications requiring guaranteed error correction performance devoid of error floors.

The theory for constructing 2D codes was first crafted by Imai [10]. Blahut [11] introduced higher dimensional finite field transform to realize algebraic codes over curves and also introduced 2D product codes in [12], capable of correcting burst errors. Each row and column of a 2D product code belong to different 1D code spaces, and the zeros of the code in frequency domain was chosen in blocks of contiguous locations. Madhusudhana and Siddiqui [13] extended the Blahut decoding algorithm for 2D BCH codes where it was required to have a block of contiguous positions set to zero in the frequency domain, and the syndromes are calculated sequentially. They showed correction of random and burst errors from a deconvolution viewpoint. The decoding algorithm for correcting random errors for 2D codes was presented by Sakata and others in [14], [15]. Sakata extended the Berlekamp-Massey decoding algorithm for 1D Reed-Solomon codes to 2D. The algorithm obtains the minimal basis of a bi-variate ideal. The error locations are component-wise inverse of the common roots of these basis polynomials. However, in Sakata's work the code construction is not explicit and does not have a mechanism for correcting multiple error patterns. Buchberger [16] developed an alternative method to obtain the minimal basis of a bi-variate ideal where, given a set of generator polynomials for a bi-variate ideal, the algorithm computes another set of generator polynomials which forms the minimal basis of that particular bi-variate ideal. The Köetter algorithm [17] provides an alternative structure to the Sakata algorithm by enforcing a regular structure but suffers from extra computations unlike the Sakata algorithm. All the aforementioned works do not consider any specific burst error pattern. Yoon and Moon designed a code in [18] for correcting a single occurrence of an error pattern from a specified set of predefined error patterns. They achieved this code construction by providing a disjoint syndrome criteria among specific error patterns. In our earlier work [19], we have analyzed the same problem from a frequency domain perspective. We were able to correct single occurrence of specific burst error patterns and multiple occurrences of a single burst pattern by solving a set of syndrome equations.

In this work, we provide procedures for correcting random errors within a 2D code as long as the errors lie within the minimum distance of the code. We do not consider any predefined shape in contrast to [18] thus, do not consider the disjoint syndrome criteria. This relaxes the constraints in choosing the set of common zeros [10]. In contrast to [12], the parity frequencies are not taken in contiguous locations. The frequency domain components belong to a binary extension field having multiple conjugate sets. In this work we have chosen the parity frequencies in locations corresponding to elements belonging to a particular conjugate set. It must be noted that these locations need not be contiguous. In [14], syndromes from the received codeword are used to find the error locator ideal. The inverse of the roots of these basis polynomials are identified as erroneous positions. In this work, we approach the problem in a different way. First, the syndromes obtained from the received codeword are used to construct an error space parity check tensor (ESPCT) and an error space generator tensor (ESGT). These tensors are different for different errors. The decoding algorithm uses a 2D Chien search for finding roots of all the polynomials in the ESGT followed by obtaining the common roots among all the root sets. The inverses of these roots give us the error locations. We illustrate the working of our algorithms with

examples.

The paper is organized as follows. In Section II, we introduce some mathematical preliminaries regarding 2D arrays over a finite field, along with 2D finite field Fourier transforms. In Section III, we introduce the various attributes of our proposed code. In Section IV, we introduce a novel 2D Chien search algorithm. In Section V, we introduce our proposed decoding algorithms, followed by conclusions in Section VI.

II. Mathematical Preliminaries

A. Two dimensional arrays over a finite field

We begin this section with a few definitions pertaining to bi-variate polynomials over a finite field. Let F be a two-dimensional array of size $n_1 \times n_2$ over a field $GF(q)$. Let $f(x, y)$ be the polynomial expansion of this array as shown below

$$f(x,y) = \sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} f_{i,j} x^i y^j \tag{1}$$

where, $f_{i,j} \in GF(q)$.

Bi-index : Every coefficient $f_{i,j} \in GF(q)$ has an index of the form (i, j) called the *bi-index*, corresponding to a monomial.

Ordering of monomials and total order: In order to identify each term of a bi-variate polynomial sequentially, it is necessary to arrange all the monomials in a particular order. The bi-indices (k_i, l_i) and (k_j, l_j) of a bi-variate polynomial are said to be in total order i.e., $(k_i, l_i) \preceq (k_j, l_j)$ if the following holds true.

$$k_i < k_j \text{ and } l_i < l_j \text{ or, } k_i = k_j \text{ and } l_i = l_j \tag{2}$$

Bi-degree: The bi-index appearing last within a total order is defined as the bi-degree of the polynomial.

Root set of a bi-variate polynomial : A point (x', y') is said to be a root of $f(x, y)$ if $f(x', y') = 0$. If $f_{i,j} \in GF(q)$, the roots of the polynomial will belong to the extension field [10] $GF(q^\lambda)$, where λ is an integer such that

$$q^\lambda = \text{lcm}(n_1, n_2) + 1, \tag{3}$$

where $\text{lcm}(.)$ denotes the least common multiple. All roots of $f(x, y)$ belong to $GF(q^\lambda)$. Let α be the primitive element of $GF(q^\lambda)$. All elements of $GF(q^\lambda)$ are of the form α^μ where $0 \le \mu \le q^\lambda - 2$ with $\alpha^{q^{\lambda-1}} = 1$. Let the maximum degree that x and y can take in equation (1) be $n_1 - 1$ and $n_2 - 1$ respectively. Consider two elements γ and β belonging to $GF\left(q^\lambda\right)$ with orders n_1 and n_2 and is assumed to be relatively prime.Thus, we get, $q^\lambda = n_1 n_2 + 1$. With α as a primitive element of $GF(q^\lambda)$, we have $\alpha^{q^\lambda - 1} = 1$. This shows $\alpha^{n_1 n_2} = 1$ and we get, $\gamma = \alpha^{n_2}$ and $\beta = \alpha^{n_1}$. Any bi-variate polynomial whose bi-degree is $(n_1 - 1,\ n_2 - 1)$ over an extension field $GF(q^\lambda)$ satisfying equation (3) has roots from the set

$$f_{\text{roots}} := \{(\gamma^i, \beta^j) | 0 \le i \le n_1 - 1 \text{ and } 0 \le j \le n_2 - 1\}. \tag{4}$$

Throughout this paper we deal with binary codewords ($q = 2$).

B. 2D Finite Field Fourier Transforms

The 2D finite field Fourier transform (FFFT) for a 2D array of dimension $n_1 \times n_2$ having elements from $GF(q)$ is defined as

$$F_{\theta,\phi} = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} f_{i,j} \gamma^{i\theta} \beta^{j\phi}. \tag{5}$$

$F_{\theta,\phi} \in GF(q^\lambda)$, such that $q^\lambda = \text{lcm}(n_1, n_2) + 1$. The 2D finite field inverse finite field Fourier transform (IFFFT) is defined as

$$f_{i,j} = \frac{1}{\text{mod}(n_1, p)} \frac{1}{\text{mod}(n_2, p)} \sum_{\theta=0}^{n-1} \sum_{\phi=0}^{m-1} \gamma^{-i\theta} \beta^{-j\phi} F_{\theta,\phi}, \tag{6}$$

where, p is the characteristic of the field and $\text{mod}(n_1, p)$ is defined as the remainder obtained while dividing n_1 with p.

III. Design of 2D Binary Linear Codes

A 2D binary code $\mathscr{C}$ of dimension $n \times m$ is a collection of arrays having n rows and m columns having elements from $GF(2)$. A 2D array can be written as a bi-variate polynomial

$$c(x,y) = \sum_{(i,j)\in\Omega} c_{i,j} x^i y^j, \tag{7}$$

where, $\Omega = \{(i, j) | 0 \le i \le n - 1, 0 \le j \le m - 1\}$ and $c_{i,j} \in GF(2)$. A 2D code is linear if $\sum_i c_i(x, y) \in \mathscr{C}$, $\forall c_i(x, y) \in \mathscr{C}$. Let $\mathscr{C}[\Omega]$ denote the set of all binary bi-variate polynomials whose maximum bi-degree of x and y are $n - 1$ and $m - 1$ respectively.

A. Code description

A random error is defined as the subset of arbitrary collection of coordinates from Ω. While decoding, each erroneous coordinate is decoded individually. This is our *error model.*

Every codeword in the codespace is a bi-variate polynomial of size $n \times m$. We have chosen, n and m to be odd and relatively prime. Let, $\gamma, \beta \in GF(2^\lambda)$ with orders n and m respectively and λ is chosen such that $2^\lambda = nm + 1$.The roots of the polynomials belong from the set $V = \left\{\left(\gamma^i, \beta^j\right) | 0 \le i \le n - 1, 0 \le j \le m - 1\right\}$. We choose a subset $V_c \subseteq V$ which forms the common roots to all the codewords in our codespace which indicates that the codespace forms an ideal which vanishes at the roots belonging to V_c. The common roots are also referred to as common zeros. The frequency domain component from equation (5), $C_{\theta,\phi} \in GF(2^\lambda)$ where λ is chosen such that $2^\lambda = nm + 1$. This condition poses a constraint on the code dimension. We illustrate this in Example 1.

Example 1: If the code is over $GF(2)$ i.e., choosing $q = 2$, the code dimensions could be chosen as follows. With $n = 3$ and $m = 5$, the frequency components $C_{\theta,\phi} \in GF(2^4)$. However, with $n = 5$ and $m = 7$, although relatively prime, this is not possible, since there does not exist a λ for which $2^\lambda = \text{lcm}(5, 7) + 1$. Another example could be $n = 7$ and $m = 9$ and so on. The code geometry is rectangular of odd area. However, we can also choose a code whose structure is a square with odd area. Let $n = m = 127$. In this case we

consider the field $GF\left(2^7\right)$. Let α be the primitive element of $GF\left(2^7\right)$ such that $\alpha^{127}=1$. From equation (5), the FFFT translates to

$$C_{\theta,\phi}=\sum_{i=0}^{126}\sum_{j=0}^{126}\alpha^{(i\theta+j\phi)}c_{i,j}. \tag{8}$$

for $c(x,y)=\sum_{i=0}^{126}\sum_{j=0}^{126}c_{i,j}x^iy^j$ where $c_{i,j}\in GF(2)$ and $C_{\theta,\phi}\in GF(2^7)$.

Parity positions: $|V_c|$ is equal to the number of parity bits in the code. The position of the parity bits in a systematic form was first discussed in [10]. The conjugacy constraint over the elements in V_c is used to define the parity positions. For any set of zeros V_c common to some arbitrary 2D polynomials over $GF(2)$, if $(\delta,\sigma)\in\Delta$, then $\left(\delta^{2^k},\sigma\right)\in\Delta$ for $k=1,2,...,\mu-1$, where μ is the least positive integer for which $\delta=\delta^{2^\mu}$. The common zeros over a set of 2D binary arrays are split using an equivalence relation to get the conjugate sets. According to [10], $\Delta_f^{(i)}$ is defined as the distinct first components in the ith conjugate set Δ_i with $\Phi_f^{(i)}:=\frac{|\Delta_i|}{\Delta_f^{(i)}}$. Let us look into the following example.

Example 2: Let us consider a set of binary arrays of size 3×5 with $\gamma^3=1$ and $\beta^5=1$ and having the following set of common zeros.

$$V_c = \left\{(1,1),(\gamma,\beta),\left(\gamma,\beta^4\right),\left(\gamma^2,\beta^2\right),\left(\gamma^2,\beta^3\right)\right\} \tag{9}$$

The common zero set is split into equivalence classes comprising of a set of conjugate roots. (δ,σ) and (η,ϕ) are said to be in an equivalence class if $\eta=\delta^{2^k}$ for $k=1,2,...,\mu-1$ and μ is the least positive integer for which $\delta=\delta^{2^\mu}$. With this condition, we have the following set of equivalence classes.

1) $\Delta_1=\left\{(\gamma,\beta),\left(\gamma,\beta^4\right),\left(\gamma^2,\beta^2\right),\left(\gamma^2,\beta^3\right)\right\},\Delta_f^{(1)}=2,\Phi_f^{(1)}=2.$
2) $\Delta_2=\{(1,1)\},\Delta_f^{(2)}=1,\Phi_f^{(2)}=1.$

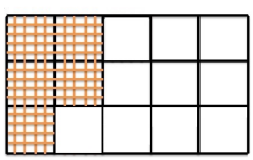

Figure 1. Parity positions of the 2D code choosing conjugacy constraint on the first component. The parity bits are all organized in contiguous positions within the shaded square. The corresponding coordinates in the parity check tensor of the code are naturally linearly independent over $GF(2)$.

A 2D code is completely characterized by the common zeros. The number of message positions is simply $k=nm-|V_c|$.

Minimum distance of the code: The size of the code is $n\times m$, and the number of parity bits is equal to $|V_c|$. From the Singleton bound, the minimum distance is given by

$$d_{\min}\leq|V_c|+1. \tag{10}$$

With equality, we would have a maximum distance separable (MDS) code.

Correction radius and cardinality of the common zero set of the code: The error correction radius is a design parameter. Every other parameter of the code is distilled after choosing the correction radius. The optimum cardinality of V_c for a $t-$error correcting code is obtained after we select the correction radius. We will derive the condition from the Hamming and Gilbert-Varshamov bounds. For a binary code of size $n\times m$ with k message bits, using the Hamming bound, we get

$$|\mathscr{C}|\leq\frac{2^{nm}}{\sum_{z=0}^{t}\binom{nm}{z}}. \tag{11}$$

Putting $|\mathscr{C}|=2^k$ with $k=nm-|V_c|$, where k is the number of message bits, we get

$$|V_c|\geq\left\lceil\log_2\left\{\sum_{z=0}^{t}\binom{nm}{z}\right\}\right\rceil. \tag{12}$$

Similarly, from Gilbert-Varshamov bound, we get

$$|V_c|\leq\left\lfloor\log_2\left\{\sum_{z=0}^{d_{\min}-1}\binom{nm}{z}\right\}\right\rfloor. \tag{13}$$

Since it is well known that $d_{\min}\geq 2t+1$, we get, $\sum_{z=0}^{t}\binom{nm}{z}\leq\sum_{z=0}^{d_{\min}-1}\binom{nm}{z}$. Thus, the optimal choice of $|V_c|$ is given by

$$\left\lceil\log_2\left\{\sum_{z=0}^{t}\binom{nm}{z}\right\}\right\rceil\leq|V_c|\leq\left\lfloor\log_2\left\{\sum_{z=0}^{d_{\min}-1}\binom{nm}{z}\right\}\right\rfloor \tag{14}$$

Choice of common zeros: In our earlier work [19] we have shown that the encoding of 2D binary codewords is done using the following conjugacy constraint [12].

$$C_{\theta,\phi}^2=C_{(\text{mod}(2\theta,n)),(\text{mod}(2\phi,m))}. \tag{15}$$

However, the work done in [12] uses this constraint for encoding a 2D code using two product codes. The parity frequencies of the 2D code are chosen in blocks of contiguous locations. The parity frequencies of our code are taken only over conjugate classes of the extension field. We explain this idea after proving the Proposition 1.

Proposition 1: The size of the largest conjugate set of the extension field $GF(2^k)$ is k.

Proof: Let α be the primitive element of $GF(2^k)$. Thus, the non-zero elements of the field are of the form α^μ for $0\leq\mu\leq2^k-2$ with $\alpha^{2^k-1}=1$. Now let us take one element α^μ. In a binary extension field, for an element α^μ, the elements of the conjugate classes are of the from $\left(\alpha^\mu\right)^{2^\delta}$, where the smallest value of $\delta=0$. We have to prove that the largest value of δ is $k-1$. In other words, if $\delta=k$, we will circle back to α^μ. The elements of the conjugate class to which α^μ belongs can be re-written in the form $\alpha^{\text{mod}\left(2^\delta\mu,2^k-1\right)}$. Thus, it is enough to show that $\alpha^{\text{mod}\left(2^k\mu,2^k-1\right)}=\alpha^\mu$ for $0\leq\mu\leq2^k-2$.

Now, for $\mu=0$ we have the trivial case which is the conjugate class with only 1 element i.e., α^0. For $\mu>0$, we have, $2^k\mu>2^k-1$. It is quite obvious that the remainder left after dividing $2^k\mu$ with 2^k-1 is μ. Thus, mod $\left(2^k\mu,2^k-1\right)=\mu$. Hence the largest value that δ can take is $k-1$. Thus, the

range of values of δ is $0 \leq \delta \leq k-1$, which denotes the maximum number of elements in a conjugate set of $GF(2^k)$ is k. ■

Let us assume we have chosen one coordinate in the frequency domain as our parity frequency. According to the conjugacy constraint of equation (15), all the coordinates which belong to the conjugate class of the chosen coordinate become parity frequencies automatically. The Figure 2 shows the conjugate classes for a 3×5 code.

1	2	2	2	2
3	4	5	5	4
3	5	4	4	5

Figure 2. The frequency domain elements of a 3×5 code belongs to the extension field $GF(2^4)$. The cells with the same numbers belong to the same conjugate classes.

Suppose for the above code we want five parity frequencies. We can choose the conjugate classes 1 and 2, or classes 1 and 4, or conjugate classes 1 and 5. It is clear that the spectral nulls are not always chosen in contiguous locations unlike in [13].

Lemma 1: If $\left(\gamma^{\theta}, \beta^{\phi}\right)$ is a zero of the code space, then in the frequency domain $C_{\theta,\phi} = 0$.

We provide the proof of the above lemma and details of the encoding procedure in our earlier work in [19].

IV. 2D Chien Search

In this section we will derive the 2D Chien search procedure. The procedure takes a bi-variate polynomial $\Lambda(x, y)$ over $GF(2)$ as its input and finds all elements of the form $\left(\gamma^k, \beta^l\right)$ such that $\Lambda\left(\gamma^k, \beta^l\right) = 0$, where $\gamma, \beta \in GF(2^{\lambda})$. For a brute force method to find the roots of any polynomial $f(x, y)$ defined over some finite field, we need to evaluate it over all the field elements one-by-one. This can be quite demanding and inefficient for large code sizes.

The 1D Chien search algorithm in [20] can be extended to 2D for finding roots of any bi-variate polynomial. Consider a bi-variate polynomial $\Lambda(x, y)$ having roots from the set $\Lambda_{\text{roots}} := \{(\gamma^i, \beta^j) | 0 \leq i \leq n-1 \text{ and } 0 \leq j \leq m-1\}$ with maximum degree for x and y to be $n-1$ and $m-1$ respectively.

$$\Lambda(x, y) = \sum_{\sigma=1}^{t} \Lambda_{k_\sigma, l_\sigma} x^{k_\sigma} y^{l_\sigma}. \tag{16}$$

Consider the following equations obtained by evaluating the bi-variate polynomial in (16) for four different roots.

$$\Lambda(\gamma^i, \beta^j) := \sum_{\sigma=0}^{t-1} \eta_{k_\sigma, l_\sigma, i, j}, \tag{17}$$

$$\Lambda(\gamma^i, \beta^{j+1}) := \sum_{\sigma=0}^{t-1} \eta_{k_\sigma, l_\sigma, i, (j+1)},$$

$$\Lambda(\gamma^{i+1}, \beta^j) := \sum_{\sigma=0}^{t-1} \eta_{k_\sigma, l_\sigma, (i+1), j},$$

$$\Lambda(\gamma^{i+1}, \beta^{j+1}) := \sum_{\sigma=0}^{t-1} \eta_{k_\sigma, l_\sigma, (i+1), (j+1)}. \tag{18}$$

The following key 2D recursive relations follow from equations (17)-(18) for $0 \leq \sigma \leq t-1$.

$$\begin{aligned}
\eta_{k_\sigma, l_\sigma i, (j+1)} &= \eta_{k_\sigma, l_\sigma, i, j} \times \beta^{l_\sigma}. && (19)\\
\eta_{k_\sigma, l_\sigma, (i+1), j} &= \eta_{k_\sigma, l_\sigma, i, j} \times \gamma^{k_\sigma}. && (20)\\
\eta_{k_\sigma, l_\sigma, (i+1), (j+1)} &= \eta_{k_\sigma, l_\sigma, i, j} \times \gamma^{k_\sigma} \beta^{l_\sigma}, \\
&= \eta_{k_\sigma, l_\sigma, (i+1), j} \times \beta^{l_\sigma}, \\
&= \eta_{k_\sigma, l_\sigma i, (j+1)} \times \gamma^{k_\sigma}. && (21)
\end{aligned}$$

Fixing an initial condition at (γ^0, β^0) given by $\eta_{k_\sigma, l_\sigma, 0, 0} = \Lambda_{k_\sigma, l_\sigma}$, we can evaluate the rest of the values using equations (19)-(21). We finally obtain, (γ^i, β^j) is a root of $\Lambda(x, y)$ if $\sum_{\sigma=0}^{t-1} \eta_{k_\sigma, l_\sigma, i, j} = 0$.

Complexity calculation

Consider a monomial $x^{k_\sigma} y^{l_\sigma}$ in $\Lambda(x, y)$ of equation (16). The number of multiplications to evaluate this monomial is $(k_\sigma - 1) \times (l_\sigma - 1)$. An additional multiplication takes place to multiply the coefficient associated with the monomial. The time complexity due to additions is $\tau_A^{(b)} = nm \times (t-1)$. Assuming the multiplication operation of all the terms is happening in parallel, the worst case complexity will occur if one of the monomials is $x^n y^m$. Thus, the multiplication time complexity is $\tau_M^{(b)} = nm \times \{(n-1) \times (m-1) + 1\}$. The total complexity considering all possible roots is

$$\tau^{(b)} = nm \times \{((n-1) \times (m-1) + 1) + t - 1\}. \tag{22}$$

But, if the multiplication operation in each term is occurring sequentially, the worst case scenario will arise if all possible monomial terms are present. The total multiplication time will be the sum of all the multiplication times for each monomial. The time for addition remains the same. To calculate the time complexity using the 2D Chien search algorithm, consider Table I.

Table I
Iterative nature of the 2D Chien Search

Roots	Coefficients		
(γ^0, β^0)	$\eta_{k_0, l_0, 0, 0}$	$\cdots$	$\eta_{k_{t-1}, l_{t-1}, 0, 0}$
(γ^0, β^1)	$\eta_{k_0, l_0, 0, 1}$	$\cdots$	$\eta_{k_{t-1}, l_{t-1}, 0, 1}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$(\gamma^{n-1}, \beta^{m-1})$	$\eta_{k_0, l_0, n-1, m-1}$	$\cdots$	$\eta_{k_{t-1}, l_{t-1}, n-1, m-1}$

The contents of Table I are populated using equations (19)-(21). The first row is obtained by $\eta_{k_\sigma, l_\sigma, 0, 0} = \Lambda_{k_\sigma, l_\sigma}$. The number of additions in each row is $(t-1)$. The values of the kth row are obtained by multiplying each element of $(k-1)^{\text{st}}$ row by a single value. Hence, the total number of multiplications in each row is t. Thus, the total number of additions is $\tau_A^{(c)} = mn \times (t-1)$. In total, we have $\tau_M^{(c)} = (mn-1) \times t$ multiplications, yielding a total time complexity

$$\tau^{(c)} = mn \times (t-1) + (mn-1) \times t. \tag{23}$$

This shows that the complexity of the 2D Chien search algorithm is $\mathcal{O}(nm)$ in contrast to $\mathcal{O}(n^2 m^2)$ from the brute force approach.

Further reduction of computational complexity

As discussed before, we have shown that the computational complexity is linear in maximum polynomial bi-degree. In this section we will use the conjugacy property of a finite field to reduce the complexity even further. Consider an extension field $GF(2^\lambda)$. Let $\nu_1, \nu_2 \in GF(2^\lambda)$. ν_2 is said to be a conjugate of ν_1 if $\nu_2 = \nu_1^{2^l}$ for any $l \geq 0$. If an element of a finite field $GF(2^\lambda)$ is a root of a polynomial over $GF(2)$, then all its conjugate pairs are also the roots of the same polynomial. All the elements of a finite field are segregated into equivalence classes according to the conjugacy constraint [21].

Consider the elements $\alpha, \gamma, \beta \in GF(2^\lambda)$, where α is primitive element, and γ and β are elements with orders n and m respectively. We assume that n and m are relatively prime. Any non-zero element of $GF(2^\lambda)$ can be written as α^μ for $0 \leq \mu \leq 2^\lambda - 2$. Since n and m are relatively prime, $\alpha^\mu = \gamma^k \beta^l$, where the pair (k, l) with $0 \leq k \leq n-1$, $0 \leq l \leq m-1$ is unique for a given μ.

Fact 1: Let $\alpha^{\mu_1}, \alpha^{\mu_2} \in GF(2^\lambda)$. Let $\gamma, \beta \in GF(2^\lambda)$ with orders n and m respectively such that n and m relatively prime. Let $\alpha^{\mu_1} = \gamma^{k_1}\beta^{l_1}$ and $\alpha^{\mu_2} = \gamma^{k_2}\beta^{l_2}$ with $0 \leq k_1, k_2 \leq n-1$ and $0 \leq l_1, l_2 \leq m-1$. If α^{μ_1} and α^{μ_2} are conjugate pairs, then $\left(\gamma^{k_1}, \beta^{l_1}\right)$ and $\left(\gamma^{k_2}, \beta^{l_2}\right)$ are also conjugate pairs through a bijection.

The proof of the above fact is straightforward since there is a bijection between α^{μ_a} and $\left(\gamma^{k_a}, \beta^{l_a}\right)$, implying, conjugate elements α^{μ_1} and α^{μ_2} must have corresponding conjugate bi-indices.

Example 3: Consider the field $GF(2^4)$ with α as the primitive element over the minimal polynomial $x^4 + x + 1$. Let $\gamma, \beta \in GF(2^\lambda)$ have orders 3 and 5 respectively. Consider an element, say α^{14}. Following the details in Section II, we find $\alpha^{14} = \gamma^1\beta^3$, corresponding to the entry in location $(1, 4)$ within a 3×5 array. Table II outlines all the elements of $GF(2^4)$ at each (k, l) coordinate in a similar fashion.

Table II
EXPRESSING $\gamma^k\beta^l$ FOR $0 \leq k \leq 2$ AND $0 \leq l \leq 4$ IN A 3×5 ARRAY.

1	α^3	α^6	α^9	α^{12}
α^5	α^8	α^{11}	α^{14}	α^2
α^{10}	α^{13}	α	α^4	α^7

All the non-zero elements of $GF(2^4)$ are split into equivalence classes following the conjugacy constraint. From Fact 1, the bi-variate elements of the form $\left(\gamma^k, \beta^l\right)$ for $0 \leq k \leq 2$ and $0 \leq l \leq 4$ also fall into their corresponding conjugate classes.

Table III
CONJUGATE CLASSES

Conjugate classes of $GF(2^4)$	Bi-variate conjugate classes
1	$\{(\gamma^0, \beta^0)\}$
$\{\alpha,\ \alpha^2,\ \alpha^4,\ \alpha^8\}$	$\{(\gamma^2, \beta^2), (\gamma^1, \beta^4), (\gamma^2, \beta^3), (\gamma^1, \beta^1)\}$
$\{\alpha^3,\ \alpha^6,\ \alpha^9,\ \alpha^{12}\}$	$\{(\gamma^0, \beta^1), (\gamma^0, \beta^2), (\gamma^0, \beta^3), (\gamma^0, \beta^4)\}$
$\{\alpha^5,\ \alpha^{10}\}$	$\{(\gamma^1, \beta^0), (\gamma^2, \beta^0)\}$
$\{\alpha^7,\ \alpha^{11},\ \alpha^{13},\ \alpha^{14}\}$	$\{(\gamma^2, \beta^4), (\gamma^1, \beta^2), (\gamma^2, \beta^1), (\gamma^1, \beta^3)\}$

Proposition 2: The 2D Chien search procedure terminates in exactly $N_2^{(\lambda)}$ steps where $N_2^{(\lambda)} = \sum_{\sigma=1}^{\lambda}\left(\frac{1}{\sigma}\sum_{d|\sigma}\mu(d) \times 2^{\frac{\sigma}{d}}\right)$ and $\mu(d)$ is the Möbius function [23].

Proof: From Fact 1 and Table III, we have already established that the bi-variate roots are uniquely mapped to the elements of an extension field and can be partitioned into conjugate classes. Consider a polynomial over $GF(2)$ which has its roots from the extension field $GF(2^\lambda)$. It is a well known result in algebra [21] that, if an element of a conjugate class satisfies the polynomial, the remaining elements of the conjugate class will also satisfy the same polynomial. Thus, the entire complexity of the proposed 2D Chien search procedure can be reduced by evaluating the roots only at one root per conjugate set. Thus, the problem reduces to counting the number of conjugate classes in the extension field $GF(2^\lambda)$. However, we know every conjugate set has its own unique minimal polynomial. Thus, counting the number of minimal polynomials in an extension field will suffice.

In a binary extension field $GF(2^\lambda)$, the maximum number of elements that can exist in one conjugate class is λ. Thus, the maximum degree a minimal polynomial in $GF(2^\lambda)$ can have is λ. From [23], the number of minimal polynomials of degree σ where $1 \leq \sigma \leq \lambda$ in $GF(2^\lambda)$ is $N(\sigma) = \frac{1}{\sigma}\sum_{d|\sigma}\mu(d) \times 2^{\frac{\sigma}{d}}$ where, $\mu(d)$ is the Möbius function [23]. Thus, the total number of minimal polynomial in $GF(2^\lambda)$ is given by

$$N_2^{(\lambda)} = \sum_{\sigma=1}^{\lambda}\left(\frac{1}{\sigma}\sum_{d|\sigma}\mu(d) \times 2^{\frac{\sigma}{d}}\right). \tag{24}$$

Thus, the complexity is reduced, and the procedure terminates in exactly $N_2^{(\lambda)}$ steps. ■

Proposition 2 helps us to evaluate the roots only at one element per conjugate set. This is a *significant* reduction in the complexity. The coefficients of $\Lambda(x, y)$ are first entered into the algorithm and added to check if the sum is zero or not. The coefficients of $\Lambda(x, y)$ are then multiplied with one root per conjugate set. The recursive nature of the 2D Chien search algorithm is performed with those chosen elements and the coefficients are updated according to equations (19)-(21).

V. DECODING

Random errors is defined as a arbitrary collection of erroneous locations. The erroneous locations may be scattered across the code or may form an arbitrary shape. Since the erroneous locations do not have a specific shape, it requires sophisticated decoding algorithms to decode each of erroneous locations individually.

In 1D, the Berlekamp-Massey algorithm helps us to find an error locator polynomial from a given set of syndrome values. The inverse of the roots of this polynomial indicate the error locations. In a 1D polynomial, the degree of the polynomial indicates the maximum number of distinct roots possible. The degree of the error locator polynomial is indicative of the error weight. The situation is not so simple in 2D. In a 2D polynomial, unlike 1D, the bi-degree of the polynomial does not have any direct indication on the number of roots. For

Algorithm 1 2D Chien Search

Input:

1) Bi-variate polynomial $\Lambda(x,y) = \sum_{\sigma=0}^{t-1} \Lambda_{k_\sigma,l_\sigma} x^{k_\sigma} y^{l_\sigma}$.
2) Maximum possible bi-degree of the polynomial i.e., n and m. [n and m are assumed to be relatively prime]

Define:

1) C_r :=array containing one conjugate root per conjugate set.
2) R := array to store bi-variate roots of $\Lambda(x,y)$.

Step 1:

1) Set $\lambda \leftarrow \log_2(nm+1)$.

Step2:

1) Obtain the elements of the extension field $GF(2^\lambda)$ in an ordered sequence and partition into conjugate classes.
2) $C_r[k]$ ←one root of conjugate set k.

Step 3:

For $k = 1$ to $N_2^{(\lambda)}$

1) Initialize row 1 of Table 1 using $\eta_{k_\sigma,l_\sigma,0,0} = \Lambda_{k_\sigma,l_\sigma}$.
2) Choose the root from $C_r[k]$ and update Table 1 using equations (19)-(21).
3) If the root at $C_r[k]$ evaluates to zero, enter all roots corresponding to conjugate set k in R; else, disregard the conjugate set k completely.
4) $k \leftarrow k+1$.

End For

Output:

1) Roots of the polynomial $\Lambda(x,y)$ stored in R.

the 2D case, we need to find a error locator ideal which is a collection of bi-variate polynomials. The component wise inverse of the common roots in the bi-variate ideal indicates the error locations. Sakata's algorithm [14], [15] is the extension of the Berlekamp-Massey algorithm to 2D where he provides a procedure for computing a set of connection polynomials for a bi-variate ideal from an ordered sequence of bi-variate monomials. The Sakata algorithm is an iterative algorithm. At the end of the process, the connection polynomials form the minimal polynomials as required in the error locator ideal [11], [14]. The number of polynomials obtained at each iteration are not the same. The number of iterations are equal to the number of terms in the syndrome polynomial, in turn, depending on the cardinality of the common zero set. But, the operations carried out in each step are computationally intensive. The Köetter algorithm [17] provides a different construction yielding a systematic structure. In contrast to the Sakata algorithm, the number of connection polynomials [11] in each step of the Köetter algorithm are equal. This algorithm offers the advantage for a systolic implementation. For any bi-variate ideal over $GF(q)$, there are at most q possible polynomials in the minimal basis of the ideal. The Köetter algorithm iterates each step with these q polynomials. The algorithm creates q copies of the Berlekamp Massey algorithm on the same syndrome polynomial to resemble the Sakata algorithm. The syndromes are first computed from the received codeword and subsequently used for obtaining the error locator ideal.

In our decoding algorithm we have developed techniques to obtain the error locator ideal in *one step* from the syndromes obtained from the received codeword. We then compute the common roots of the polynomials in the error locator ideal by using the 2D Chien search. The error positions are componentwise inverse of these common roots. Before we delve into the details of the decoding procedure, we are going to present some notations, useful henceforth.

Notations and Definitions:

- Let the error polynomial be $e(x,y) = \sum_{i=0}^{\lambda-1} e_{i_\sigma,j_\sigma} x^{i_\sigma} y^{j_\sigma}$, where $e_{i_\sigma,j_\sigma} \in GF(2)$ and has λ error locations.
- $e(x,y)$ as a 2D binary matrix is denoted by E and its FFFT is denoted by $\hat{E}$.
- $\Omega = \{(i,j) | 0 \le i \le n-1 \text{ and } 0 \le j \le m-1\}$. This defines all the coordinates in a 2D matrix of size $n \times m$.
- E_r denotes the set of error locations, i.e., $E_r = \{(i_1,j_1), (i_2,j_2), \cdots, (i_\lambda,j_\lambda)\}$ and $|E_r| = \lambda$.
- V_c is the set of common zeros for the code space in the time domain. $|V_c| = \chi$ and $\lambda < \chi$.
- VF_c is the set of zeros in the frequency domain. $VF_c = \left\{(\theta_\zeta, \phi_\eta) \,|\, \left(\gamma^{\theta_\zeta}, \beta^{\phi_\eta}\right) \in V_c\right\}$.
- S denotes the syndromes evaluated on the received codeword at the common zero positions.
- S_L denotes the set of linearly independent syndrome components and $S_L \subseteq S$.
- $E_z := \left\{(k,l) \,|\, \gamma^k \beta^l \in S_L\right\}$ and $|E_z| = \chi$. $\hat{E}_z = \Omega - E_z$.

The erroneous coordinates of a received codeword are component-wise inverse of the common roots of a set of bi-variate polynomials. These set of bi-variate polynomials form an ideal [11]. Our primary task is to find this ideal from the syndromes obtained from the received codeword.

Example 4: Consider the field $GF(2^4)$ with primitive element α over the minimal polynomial x^4+x+1. Consider two other elements γ and β with orders 3 and 5 respectively from the same field. We have

$$\alpha^{2^4-1} = 1. \tag{25}$$

Using $\gamma = \alpha^5$ and $\beta = \alpha^3$, and the fact that $\gcd(3,5) = 1$, we can express γ and β in terms of α. After receiving a codeword, the first task is to evaluate the received codeword at the common zero locations to obtain the syndromes. For every element in V_c evaluated over the received codeword, we obtain a syndrome forming the set S. Let us consider one of the syndromes $\alpha^{11} \in GF\left(2^4\right)$ that can be expressed as $\alpha^{11} = \gamma^2\beta$. Thus, the syndrome component maps to the coordinate $(1,2)$, an element in the set E_z.

The following lemma shows the uniqueness of this mapping when the code dimensions are relatively prime.

Lemma 2: The mapping from the syndrome set S_L to E_z is unique when the code dimensions are relatively prime.

Proof: For a code of size $n \times m$ over $GF(2)$, the syndromes for an erroneous binary received vector are from an extension field $GF(2^\lambda)$ where 2^λ divides $\text{lcm}(n,m)+1$. We choose the field in such a way that n and m are relatively prime. Let α be a primitive element of $GF(2^\lambda)$. Let γ and β

also be elements of $GF\left(2^{\lambda}\right)$ of orders n and m respectively. This implies $\gamma^k\beta^l \in GF(2^{\lambda})$ for $0 \leq k \leq n-1$, $0 \leq l \leq m-1$. Since $\alpha \in GF(2^{\lambda})$ is a primitive element, we have $\alpha^{2^{\lambda}-1} = 1$. Since we choose n and m to be relatively prime, we have $2^{\lambda} - 1 = n \times m$. With $\alpha^{n\times m} = 1$, $\gamma^n = 1$ and $\beta^m = 1$, we have $\gamma = \alpha^m$ and $\beta = \alpha^n$.

From the above equations, we can express $\gamma^k\beta^l = \alpha^{mk+nl}$. Now, every non zero element can be written in the form $\alpha^i \in GF(2^{\lambda})$ for $0 \leq i \leq 2^{\lambda}-1$, and they are all unique. This implies that we can write every element in the form

$$\alpha^{mk+nl} = \alpha^i, \tag{26}$$

where, $0 \leq k \leq n-1$, $0 \leq l \leq m-1$ and $0 \leq i \leq 2^{\lambda}-1$. Now, every syndrome is of the form α^i. We have to show that the decomposition of α^i to α^{mk+nl} is unique. Using logarithm with base α, we can reduce equation (26) to

$$mk + nl = \text{mod}(i, 2^{\lambda}-1). \tag{27}$$

Equation (27) represents a family of parallel lines depending on the value of i. There are exactly $2^{\lambda}-1$ parallel lines. For each syndrome component, we have exactly one line representing it. This implies every (k,l) pair is unique. ■

A. The Error Space Parity Check Tensor (ESPCT)

For a code of size $n\times m$ over $GF(2)$, the transformed codeword has elements over $GF(2^{\lambda})$ where $2^{\lambda} = \text{lcm}(n,m)+1$. All the syndrome components are over $GF(2^{\lambda})$. The elements of $\hat{E}_z$ are expressed as a linear combination of elements from E_z.The linear combination operation gives us the $(i,j)^{\text{th}}$ element of the ESPCT as

$$E_{H_{i,j}} = \sum_{(k,l)\in E_z} h_{k,l}^{(i,j)}\gamma^k\beta^l \text{ for } (i,j)\in\Omega. \tag{28}$$

The $(i,j)^{\text{th}}$ element in algebraic form is obtained as

$$E_{H_{i,j}}(x,y) = \sum_{(k,l)\in E_z} h_{k,l}^{(i,j)} x^k y^l. \tag{29}$$

The elements of E_z are obtained from S_L. The elements in ESPCT can be used to find the complete error spectrum.

B. The Error Space Generator Tensor (ESGT)

For correction of random errors in 1D codewords, various algorithms are used which uses an error locator polynomial whose roots are inverse of the error locations. The degree of this locator polynomial is equal to the number of errors in the transmitted codeword. However, in 2D the situation is complicated. We now require a set of polynomials to distill a set of common bi-variate roots [11]. This set of bi-variate polynomials form an ideal. The reason can be seen as follows: Consider the ring of bi-variate polynomials $(K[x,y],+,.)$ over the field $GF(2)$. Let $F \subset K$ have a set of common roots V. $(F,+)$ will form a ideal if the following properties [22] hold:

1) $(F+)$ is a subgroup of $(K,+)$.
2) $\forall x \in F,$ and $\forall y \in R$, $x.y \in I$.
3) $\forall x \in F,$ and $\forall y \in R$, $y.x \in I$.

Let us take a polynomial $f(x,y) \in F$ and let $r(x,y) \in K$. We multiply them to get the following.

$$f(x,y).r(x,y) = g(x,y). \tag{30}$$

Now consider the roots of $g(x,y)$. Selecting any root from V we get $f(x,y) = 0$, which in turn makes $g(x,y) = 0$. So, $g(x,y)$ will also have the same common roots as all the elements in F. Thus, $g(x,y) \in F$. We further check if $(F,+)$ forms a subgroup of $(K,+)$. Since the polynomials belonging to K are over $GF(2)$ we have,

1) the additive inverse of every polynomial is itself.
2) The identity element is clearly the zero polynomial.
3) The necessary and sufficient condition [22] to prove $(F,+)$ a subgroup of $(K,+)$ is to show for $f(x,y), g(x,y) \in F$, $f(x,y) + (-g(x,y)) \in F$. Here, $-g(x,y)$ denotes the additive inverse of $g(x,y)$. Hence we have

$$\begin{aligned} f(x,y) + (-g(x,y)) &= f(x,y) + g(x,y), \\ &= l(x,y). \end{aligned}$$

Substituting any roots from V, we get $f(x,y) = 0$ and $g(x,y) = 0$ making $l(x,y) = 0$ which proves $l(x,y) \in F$. This implies $(F,+)$ is a subgroup of $(K,+)$. Hence, F forms an ideal. Thus, the set of bi-variate polynomials having a set of common roots forms an ideal.

Each element of the ESGT is obtained as

$$E_{G_{i,j}}(x,y) = \sum_{(k,l)\in E_z} h_{k,l}^{(i,j)} x^k y^l + x^i y^j, \ \forall (i,j)\in\Omega. \tag{31}$$

The number of erroneous positions in a 2D code is now equal to cardinality of the common root set of the polynomials present in ESGT. All non zero polynomials in ESGT form the error locator ideal. The error locator ideal is specific to a particular error pattern. The number of erroneous positions in a 2D code is now equal to cardinality of the common root set of the polynomials present in ESGT. All the non-zero ESGT polynomials form the error locator ideal. The component wise inverse of the common roots of these polynomials indicates the error locations. This result is proved in Theorem 1.

Theorem 1: The following statements hold true:

1) If $E_{G_{i,j}}\left(\gamma^{-i_\sigma}, \beta^{-j_\sigma}\right) = 0 \ \forall (i,j)\in\Omega$, then $e(x,y) = x^{i_\sigma}y^{j_\sigma}$, indicating the location (i_σ, j_σ) to be in error.
2) For $(i_\sigma, j_\sigma) = (-1,-1)$, $E_{G_{i,j}}(x,y) = 0 \ \forall (i,j)\in\Omega$, indicating the location $(n-1, m-1)$ is in error. This location is considered as an erroneous location if the syndromes obtained after flipping the bit at $(n-1, m-1)$ is zero.

Proof: First, we prove statement 1. To get the error polynomial $e(x,y) = x^{i_\sigma}y^{j_\sigma}$, the coefficient e_{i_σ,j_σ} has to be proven equal to 1. Let us assume the contrary, i.e., $e_{i_\sigma,j_\sigma} = 0$. For clarity, we mention the ESGT expression again. We have,

$$E_{G_{i,j}}(x,y) = \sum_{(k,l)\in E_z} h_{k,l}^{(i,j)} x^k y^l + x^i y^j = E_{H_{i,j}}(x,y) + x^i y^j.$$

Putting $(x,y) = \left(\gamma^{-i_\sigma}, \beta^{-j_\sigma}\right)$, we have

$$E_{G_{i,j}}\left(\gamma^{-i_\sigma}, \beta^{-j_\sigma}\right) = E_{H_{i,j}}(\gamma^{-i_\sigma}, \beta^{-j_\sigma}) + \gamma^{-i_\sigma i}\beta^{-j_\sigma j} \quad (32)$$

Since $\left(\gamma^{-i_\sigma}, \beta^{-j_\sigma}\right)$ is a root of $E_{G_{i,j}}(x,y)$ $\forall (i,j) \in \Omega$, we have

$$E_{H_{i,j}}(\gamma^{-i_\sigma}, \beta^{-j_\sigma}) = \gamma^{-i_\sigma i}\beta^{-j_\sigma j}. \quad (33)$$

Multiplying the above equation with the frequency component of the error vector $E_{i,j}$ and summing over all $(i,j) \in \Omega$, we have

$$\sum_{(i,j)\in\Omega} \gamma^{-i_\sigma i}\beta^{-j_\sigma j} E_{i,j} = \sum_{(i,j)\in\Omega} E_{i,j} E_{H_{i,j}}(\gamma^{-i_\sigma}, \beta^{-j_\sigma}). \quad (34)$$

Consider equation (6). The terms $\mathrm{mod}(n,p)$ and $\mathrm{mod}(m,p)$ are both equal to 1 when we are considering codes over the binary field with n and m being odd. From (6), the left hand side of the above equation is equal to e_{i_σ, j_σ}. This gives us

$$e_{i_\sigma, j_\sigma} = \sum_{(i,j)\in\Omega} E_{i,j} E_{H_{i,j}}(\gamma^{-i_\sigma}, \beta^{-j_\sigma}). \quad (35)$$

The R.H.S of equation (35) can be interpreted as the Frobenius product of two matrices E_Ω and G_Ω where,

$$E_\Omega \coloneqq [E_{i,j}], (i,j) \in \Omega \quad (36)$$

and,

$$G_\Omega \coloneqq \left[E_{H_{i,j}}(\gamma^{-i_\sigma}, \beta^{-j_\sigma})\right], (i,j) \in \Omega. \quad (37)$$

Consider two matrices $A \coloneqq [a_{ij}]$ and $B \coloneqq [b_{ij}]$ with dimension $w \times v$. The inner product between two matrices of the same size is defined as the Frobenius norm as

$$< A, B >_{\mathscr{F}} = \sum_{i=1}^{w}\sum_{j=1}^{v} a_{ij} b_{ij}$$

From equation (33), we get

$$G_\Omega = \begin{bmatrix} 1 & \cdots & \beta^{-j_\sigma(m-1)} \\ \gamma^{-i_\sigma} & \cdots & \gamma^{-i_\sigma}\beta^{-j_\sigma(m-1)} \\ \vdots & \ddots & \vdots \\ \gamma^{-i_\sigma(n-1)} & \cdots & \gamma^{-i_\sigma(n-1)}\beta^{-j_\sigma(m-1)} \end{bmatrix}. \quad (38)$$

Thus,

$$e_{i_\sigma, j_\sigma} = < E_\Omega, G_\Omega >_{\mathscr{F}}, \quad (39)$$

G_Ω is the finite field Fourier transform of a bi-variate polynomial $g(x,y) = x^{-i_\sigma} y^{-j_\sigma}$ at all $(\theta, \phi) \in \Omega$ locations. Now, in equation (39) we have assumed $e_{i_\sigma, j_\sigma} = 0$ which is only possible if the E_Ω matrix is a zero matrix. This is a contradiction since E_Ω being a zero matrix assumes there is no error, indicating that the syndromes will also be zero, which is not true in our case. Hence, our desired result is proved.

We prove the second statement now. The way the ESGT polynomials are obtained, it makes $E_{G_{i,j}}(\gamma, \beta) = 0$ $\forall (i,j)$, making the location $(n-1, m-1)$ always to be in error. This means, while obtaining the error locations one-by-one, we also get the root $(\gamma^{-1}, \beta^{-1})$ as a root common to all the ESGT polynomials. This is an *artifact* of the process. For this, we perform a second syndrome check after we have obtained the erroneous locations and flip the bits in those locations. If the location $(n-1, m-1)$ was affected during transmission, the second syndrome check will result in a zero syndrome. However, if the location $(n-1, m-1)$ was not in error, we will still not get a zero syndrome on the second syndrome check. In that case, we again flip the bit at the location $(n-1, m-1)$ again. ■

The result $E_{G_{i,j}}\left(\gamma^{-i_\sigma}, \beta^{-j_\sigma}\right) = 0$ for $1 \leq \sigma \leq \lambda$ and $\forall (i,j) \in \Omega$ also helps us to find the entire error spectrum. The 2D IFFFT of the error spectrum gives the error vector directly. This result is added to the received vector to obtain the transmitted codeword. The decoding process does not require finding the erroneous positions explicitly.

Lemma 3: For a code of size $n \times m$, if we obtain $|S_L| = \lambda$ satisfying 2^λ = $\mathrm{lcm}(n,m) + 1$, then the received codeword is correctable.

Proof: Each syndrome component is obtained by evaluating the received codeword at the common zeros. The the transmitted codeword be affected by the error vector $e(x,y)$. Thus, for every $(\gamma^\theta, \beta^\phi) \in V_c$ we obtain a syndrome component $S_{\theta,\phi} = e(\gamma^\theta, \beta^\phi)$ and from equation (5), $S_{\theta,\phi} \in GF(2^\lambda)$. We also know from before, $S_L \subseteq S$. From Lemma 2, we can express any element belonging to S_L can be expressed uniquely in the form $\gamma^k\beta^l$ where $(k,l) \in \Omega$. All such (k,l) components also belong to the set E_z. We express $\gamma^i\beta^j$ $\forall (i,j) \in \Omega$ in terms of $\gamma^k\beta^l$ for $(k,l) \in E_z$ according to equation (28). If all $\gamma^k\beta^l$ for $(k,l) \in E_z$ are linearly independent and $S_L = \lambda$, it will form a basis over $GF(2^\lambda)$ and thus be able to generate all the ESPCT polynomials. This helps to decode the received codeword completely. ■

For a code of size $n \times m$, the frequency components of the codewords in the code space are from $GF\left(2^\lambda\right)$ where λ is chosen as such that 2^λ=lcm$(n,m) + 1$. Lemma 3 gives the following criteria for the correctability of a received codeword.

1) Error in a received vector will be correctable if $|S_L| = \lambda$.
2) Error in a received vector will be detectable but not correctable if $|S_L| < \lambda$. If the number of linearly independent syndromes for a code with frequency domain components from $GF(2^\lambda)$ is less than λ, it will not span the set completely; subsequently, some of the ESPCT polynomials will be incomplete. Without the ESPCT polynomials, we cannot proceed with the decoding of the code.

Lemma 3 also poses a constraint in choosing the number of common zeros of the code space. For a correctable error in a code of size $n \times m$, $|S_L| = \lambda$, where λ is such that $2^\lambda = \mathrm{lcm}(n,m) + 1$. For the correctability of an erroneous codeword, the rate should be at least

$$\mathscr{C}_{\text{rate}} \leq 1 - \frac{\lambda}{nm}. \quad (40)$$

C. Decoding Procedure

When a codeword is received, we first perform the syndrome check and obtain the set S. We obtain the linearly independent syndrome S_L from S. If the frequency domain components of the code are from $GF(2^\lambda)$, then from Lemma 3 if $|S_L|$ is less than λ we have to indicate that the error cannot be corrected.

Algorithm 2 Decoding algorithm for random errors using 2D Chien search.

Inputs: Received codeword $r(x,y)$ and V_c.

Step 1: Syndrome Check

1) **Compute** S using V_c on $r(x,y)$.
2) **If** all the syndrome values are zero
   - **Set** $c(x,y) = r(x,y)$.
   - Proceed to **Output**.

   Else
   - Proceed to **Step 2**.

Step 2:

1) **Obtain** the linearly independent syndromes from S to obtain S_L.
2) **Compute** E_z and $\hat{E}_z$.
3) **Compute** ESPCT and ESGT.

Step 3: 2D Chien search

1) **If** $E_{G_{i,j}}(x,y) \neq 0$,
   - Find the roots of $E_{G_{i,j}}(x,y)$ using the 2D Chien search.
   - Store the roots of $E_{G_{i,j}}(x,y)$ in $E_{\text{root}}^{(i,j)}$.

Step 4: Intersection of roots

- $E_{\text{root}} = \cap_{(i,j)\in(\Omega-E_z)} E_{\text{root}}^{(i,j)}$.

Step 5: Flip bits

1) Invert the elements of E_{root} component wise to find the error locations.
2) Flip the bits in the corresponding error locations.

Step 6: Artifact check

1) Syndrome check
2) **If** any one of the syndrome values are not zero,
   - **Flip** bit at location $(n-1, m-1)$.
   - Proceed to **Output**

   Else
   - Proceed to **Output**.

Output: $c(x,y)$.

Thus the error is detectable but not correctable. If $|S_L| = \lambda$ then we compute the set E_z and further generate the ESPCT and ESGT polynomials. The 2D Chien search procedure is used to find the roots of the ESGT polynomials. Let $E_{\text{root}}^{(i,j)}$ be the set of roots for the polynomial in $(i,j)^{\text{th}}$ positions of ESGT. Let E_{root} be defined as the set of common roots obtained by $E_{\text{root}} = \cap_{(i,j)\in(\Omega-E_z)} E_{\text{root}}^{(i,j)}$. The inverse of all the components in E_{root} are our error locations. Algorithm 2 enumerates the procedure.

Example 5: Here, we give a decoding example of a 3×5 code. For simplicity, we consider the all zero codeword.

Code parameters:

1) We consider $d_{\text{min}} = 3$, implying a correction radius of $t = 1$.
2) From equation (14), we can choose $|V_c| = 5$. We choose the following common zero set.

$$V_c = \left\{(1,\beta), (\gamma,1), (\gamma,\beta), \left(\gamma^2,1\right), \left(\gamma^2,\beta^2\right)\right\}.$$

Let the received codeword be

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

In polynomial form, it is written as $R(x,y) = xy^2$.

Step 1: Syndrome check

The first step in decoding is to evaluate the received polynomial at the common zero locations. We have, $S_{1,0} = \alpha^5$, $S_{0,1} = \alpha^6$, $S_{2,0} = \alpha^{10}$, $S_{1,1} = \alpha^{11}$ and $S_{2,2} = \alpha^7$. Thus the syndrome set is $S = \{S_{1,0}, S_{0,1}, S_{2,0}, S_{1,1}, S_{2,2}\} = \{\alpha^5, \alpha^6, \alpha^{10}, \alpha^{11}, \alpha^7\}$.

Step 2: Obtaining S_L and E_z.

We obtain the linearly independent subset $S_L = \{\alpha^5, \alpha^6, \alpha^{10}, \alpha^{11}\}$. Using Lemma 2, we obtain $E_z = \{(1,0), (0,2), (2,0), (1,2)\}$.

Step 3: Generation of the ESPCT and ESGT polynomials

Now, we obtain $\gamma^i\beta^j$ $\forall(i,j) \in \Omega$ as a linear combination of $\gamma^k\beta^l$ $\forall(k,l) \in E_z$. This is further used to obtain ESPCT and ESGT polynomials using equations (29) and (31) respectively. The values $h_{k,l}^{(i,j)}$ $\forall(k,l) \in E_z$ and $\forall(i,j) \in \Omega$ are populated in Table IV.

Step 4: Intersection of roots

Using the 2D Chien search procedure, we obtain the roots of $E_{G_{i,j}}(x,y)$ polynomials $\forall(i,j) \in \Omega$ given by $E_{\text{roots}} = \left\{\left(\gamma^2,\beta^3\right), \left(\gamma^1,\beta^1\right)\right\}$.

Step 5: Flip bits

This indicates that the locations $(1,2)$ and $(2,4)$ are in error. Thus, after flipping the bits at these locations, we deduce the transmitted codeword as

$$C^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}.$$

Step 6: Artifact check

As mentioned in Theorem 1, we perform the syndrome check again. It is easy to see, that the syndrome is non-zero. That means, the flip at location $(n-1, m-1)$ is wrong. Thus, we flip it back again. Finally this gives us the transmitted codeword which exactly same as the transmitted codeword.

D. Time complexity calculation

The main two steps involved in both the proposed algorithms are the creation of the ESPCT and ESGT polynomials. After obtaining the bi-variate expansion of each of the syndrome component, the polynomial at the $(i,j)^{\text{th}}$ coordinate is obtained from equation (28). This involves finding a linear combination of the syndrome components. The number of non-zero elements in an extension field $GF(2^\mu)$ is $2^\mu - 1$. Since the cardinality of the linearly independent set of syndromes that span the space of vectors in $GF(2^\mu)$ is μ, to find the linear combination at the coordinate (i,j), one can perform a linear search using $2^\mu - \mu - 1$ combinations. Assuming we have a parallel structure to obtain all the elements of the ESPCT, the total time complexity of this operation is $\tau_{lc}^{(p)} = mn(2^\mu - \mu - 1)$, where n and m are the code dimensions. One can store all the linear combinations in the

Table IV
LINEAR COMBINATIONS, ESGT AND ESPCT POLYNOMIALS AT ALL $(i,j) \in \Omega$ COORDINATES.

(i,j)	$h_{1,0}^{(i,j)}$	$h_{0,2}^{(i,j)}$	$h_{2,0}^{(i,j)}$	$h_{1,2}^{(i,j)}$	$E_{H_{i,j}}(x,y)$	$E_{G_{i,j}}(x,y)$
$(0,0)$	1	0	1	0	$x+x^2$	$x+x^2+1$
$(0,1)$	1	0	0	1	$x+xy^2$	$x+xy^2+y$
$(0,2)$	0	1	0	0	y^2	0
$(0,3)$	1	1	0	0	$x+y^2$	$x+y^2+y^3$
$(0,4)$	1	0	1	1	$x+x^2+xy^2$	$x+x^2+xy^2+y^4$
$(1,0)$	1	0	0	0	x	0
$(1,1)$	0	1	1	1	$y^2+x^2+xy^2$	$y^2+x^2+xy^2+xy$
$(1,2)$	0	0	0	1	xy^2	0
$(1,3)$	0	0	1	1	x^2+xy^2	$x^2+xy^2+xy^3$
$(1,4)$	1	1	0	1	$x+y^2+xy^2$	$x+y^2+xy^2+xy^4$
$(2,0)$	0	0	1	0	x^2	0
$(2,1)$	1	1	1	0	$x+y^2+x^2$	$x+y^2+x^2+x^2y$
$(2,2)$	0	1	0	1	y^2+xy^2	$y^2+xy^2+x^2y^2$
$(2,3)$	1	1	1	1	$x+y^2+x^2+xy^2$	$x+y^2+x^2+xy^2+x^2y^3$
$(2,4)$	0	1	1	0	y^2+x^2	$y^2+x^2+x^2y^4$

registers for computational ease. The length of each register is no more than μ.

Time complexity of Decoding procedure 1: According to equation (31), the ESGT polynomials are obtained directly from the ESPCT polynomials with only one addition. The total time complexity at this step is $\tau_{\text{ESGT}} = mn \times \tau_a^{(p)}$. Since there are mn ESGT polynomials we require mn number of additions. Hence, the time complexity is $\mathcal{O}(mn)$.

VI. CONCLUSIONS

We presented a code construction and a new decoding methods for handling 2D random errors. We have also developed a fast 2D Chien search algorithm to find the roots of the bivariate polynomials in the error locator ideal. The component wise inverse of the common roots of the error locator ideal gives us the erroneous locations. We have also devised a method to find the entire error spectrum. Our analysis shows that our decoding algorithms have better time complexities and ease of implementation compared to prior works. Our future work would be to develop a theory on 2D non-binary codes in the context of cluster errors.

ACKNOWLEDGMENT

S. S. Garani would like to thank the MeitY, grant no. MITO0101 for supporting this research, and IUSSTF, grant no. 16/2014 for the travel.

REFERENCES

[1] S. G. Srinivasa, Y. Chen, and S. Dahandeh, “A communication-theoretic framework for 2-DMR channel modeling: performance evaluation of coding and signal processing methods”, *IEEE Trans. Magn.,* vol. 50, no. 3, pp. 6-12, March 2014.

[2] R. Wood, M. Williams, A. Kavčić and J. Miles, “The feasibility of magnetic recording at 10 terabits per square inch on conventional media”, *IEEE Trans. Magn.,* vol. 45, no. 2, pp. 917- 923, Feb. 2009.

[3] H. J. Richter, “Recording on bit-patterned media at densities of 1Tb/in^2 and beyond”, *IEEE Trans. Magn.,* vol. 42, no. 10, pp. 2255-2260, Oct. 2006.

[4] H. T. Lue, S. H. Chen, Y. H. Shih, K.Y. Hsieh and C. Y. Lu, “Overview of 3D NAND flash and progress of vertical gate (VG) architecture", *IEEE* 11th *Conf. on Solid-State and Integ. Circuit. Tech.,* pp. 1-4, Oct. 2012.

[5] L. Hesselink, S. S. Orlov and M. C. Bashaw, “Holographic data storage systems”, *Proc. IEEE.*, vol. 92, pp. 1232-1280, Aug. 2004.

[6] Y. Chen and S. G. Srinivasa, “Joint self-iterating equalization and detection for channels”, *IEEE Trans. Comm.*, vol. 61, no. 8, pp. 3219-3230, Aug. 2013.

[7] C. K. Matcha and S. G. Srinivasa, "Generalized partial response equalization and data-dependent noise predictive signal detection over media models for TDMR", *IEEE. Trans. Magn.*, vol. 51, no.10, pp.1-15, Oct. 2015.

[8] C. K. Matcha and S. G. Srinivasa**,** "Joint timing recovery and signal detection for two-dimensional magnetic recording", *IEEE. Trans. Magn.*, Early access, Aug. 2017.

[9] C. K. Matcha, S. Roy, M. Bahrami, B. Vasic, and S. G. Srinivasa, "2D LDPC Codes and Joint Detection and Decoding for Two-Dimensional Magnetic Recording", *IEEE. Trans. Magn.*, vol. PP, no.99, Aug. 2017.

[10] H. Imai, “A theory of two-dimensional cyclic codes”, Academic Press, Inc., *Inf. and Control*, vol. 34, no. 1, pp. 1-21, May 1977.

[11] R. E. Blahut, “Algebraic codes on lines, planes and curves: An engineering approach”, Cambridge University Press, 2008.

[12] R. E. Blahut, "Transform techniques for error control codes," IBM Journal of research and development vol. 23, no. 3, May 1979.

[13] H. S. Madhusudhana and M. U. Siddiqui, “On Blahut’s decoding algorithms for two-dimensional BCH codes”, *IEEE Trans. Inf. Theory*, vol. 44, no. 1, pp. 358-367, Jan. 1988.

[14] S. Sakata, “Extension of the Berlekemp-Massey algorithm to N dimension”, *IEEE. Intl. Symp. Inf. Theory*, pp. 207-239, June 1988.

[15] K. Saints and C. Heegard, “Algebraic-geometric codes and multidimensional cyclic codes: A unified theory and algorithms for decoding using Gröbner Bases”, *IEEE Trans. Inf. Theory*, vol. 41, no. 6, pp. 1733-1751, Nov. 1995.

[16] B. Buchberger, “Gröbner Bases: An algorithmic method in polynomial ideal theory in multidimensional systems theory”, D. Reidel Publishing Company, pp. 184–232, 1985.

[17] R. Köetter, “A unified description of an error locating procedure for linear codes”, *Proceedings of Algebraic and Combinatorial Coding Theory*, Voneshta Voda, Bulgaria, 1992.

[18] S. W. Yoon, and J. Moon, “Two-dimensional error-pattern-correcting codes”, *IEEE Trans. Comm.*, vol. 63, no. 8, pp. 2725-2740, Aug. 2015.

[19] S. Roy and S. G. Srinivasa, “Two dimensional error-correcting codes using finite field Fourier transform”, *IEEE. Inf. Theory Works.*, pp. 119-123, Oct. 2015.

[20] R. T. Chien, “Cyclic decoding procedures for the Bose-Chaudhuri-Hocquenghem codes”, *IEEE Trans. Inf. Theory,* vol. 10, no. 4, pp. 357-363, Oct. 1964.

[21] S. Lin and D. J. Costello, “Error control coding”, Pearson Education India, 2011.

[22] J. M. Howie, “Fields, and Galois theory”, Springer, 2008.

[22] J. K. Wolf, A. M. Michelson and A. H. Levesque, “On the probability of undetected error for linear block codes”, *IEEE Trans. Comm.,* vol. COM-30, no. 2, pp. 317-324, Feb. 1982.

[23] N. Jacobson, “Basic Algebra I”, W. H. Freeman and Company, 1985.