Analog Error-Correcting Codes: Designs and Analysis

Anxiao Jiang, *Senior Member, IEEE*

***Abstract*— A new type of analog error-correcting codes (Analog ECCs) has been proposed by Roth recently. The codes can correct errors of unlimited magnitudes even though the codeword is affected not only by such errors, but also by ubiquitous noise of limited magnitudes. The codes have the potential to accelerate the widely used vector-matrix multiplication in machine learning via their implementation in nanoscale analog circuits. Several Analog ECCs, which mainly focus on correcting or detecting a single unlimited-magnitude error, have been proposed. This paper explores the analysis and constructions of Analog ECCs in multiple ways. It presents a linear-programming based algorithm that computes the m-heights of Analog ECCs efficiently, which can be used to determine the error correction/detection capabilities of the codes. It then presents a family of Analog ECCs based on permutations, and proves that the time complexity for determining the m-heights of such codes can be further reduced substantially. The analysis forms a basis for the time-complexity tradeoff between the searching of codes and the verification of their performance. The paper then presents a number of newly discovered codes based on such a search and verification process, which achieve state-of-the-art performance.**

***Index Terms*— Analog error-correcting codes, machine learning, permutation, resistive memories, vector-matrix multiplication.**

I. Introduction

MACHINE learning algorithms have found wide applications in many fields of engineering. A new type of Analog Error-Correcting Codes (Analog ECC), which has important potential applications to machine learning, has been proposed recently [1], [2]. Let $\mathcal{C}$ be a linear $[n, k]$ Analog ECC over $\mathbb{R}$. Let $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in \mathbb{R}^n$ denote a generic codeword in $\mathcal{C}$. There are two types of additive errors that can be added to a codeword by the channel: a type of *limited-magnitude errors* (LMEs), and a type of *unlimited-magnitude errors* (UMEs), defined as follows.

Let $[n\rangle$ denote the integer set $\{0, 1, \cdots, n-1\}$. Let δ and Δ be two positive real thresholds, where $\Delta > \delta > 0$. An error vector $\boldsymbol{\varepsilon} = (\varepsilon_0, \varepsilon_1, \cdots, \varepsilon_{n-1}) \in \mathbb{R}^n$ is called a *limited-magnitude error vector* (i.e., LME vector) if $\varepsilon_i \in [-\delta, \delta]$ for all $i \in [n\rangle$. Given a vector $\mathbf{e} = (e_0, e_1, \cdots, e_{n-1}) \in \mathbb{R}^n$, define its support with respective to Δ as

$$\mathrm{Supp}_{\Delta}(\mathbf{e}) = \{i \in [n\rangle \ : \ |e_i| > \Delta\}.$$

The above definition can be extended to $\Delta = 0$. Then by this definition, the ordinary support of $\mathbf{e}$ is $\mathrm{Supp}_0(\mathbf{e})$. And the Hamming weight of $\mathbf{e}$, denoted by $\mathrm{w}_H(\mathbf{e})$, is $|\mathrm{Supp}_0(\mathbf{e})|$. An error vector $\mathbf{e} = (e_0, e_1, \cdots, e_{n-1}) \in \mathbb{R}^n$ is called an *unlimited-magnitude error vector* (i.e., UME vector) of Hamming weight w if $\mathrm{w}_H(\mathbf{e}) = w$. A noisy codeword $\mathbf{y} = (y_0, y_1, \cdots, y_{n-1}) \in \mathbb{R}^n$ is the sum of the codeword $\mathbf{c} \in \mathcal{C}$ and the two error vectors $\boldsymbol{\varepsilon}$ and $\mathbf{e}$, namely, $\mathbf{y} = \mathbf{c}+\boldsymbol{\varepsilon}+\mathbf{e}$. The code is designed such that significant UMEs will be corrected.

A strong motivation for the introduction of Analog ECC is to support *vector-matrix multiplication* — a common operation in machine learning algorithms including deep learning [1], [2] — that is realized with a crossbar architecture. In the following, we introduce its application to *Analog In-Memory Computing* for deep neural networks (DNNs). DNNs have achieved significant progress for AI in recent years, covering computer vision, natural language processing, generative AI and more areas. However, the cost for their training and inference, in both time and power consumption, is also increasing substantially. A fundamental emerging technology, Analog In-Memory Computing, promises to make DNNs much more efficient in both speed and energy consumption [3], [4], [5], [6]. By storing the real-valued parameters of DNNs in nanoscale analog non-volatile memory (NVM) cells and using them directly for computing, in-memory analog computing may overcome the "von Neumann bottleneck" of conventional computers. The new paradigm avoids the movement of massive amounts of data between GPUs and external memories, which incurs massive energy consumption and accounts for extensive latency [7] in current AI systems. There has been good progress in the development of analog chips in recent years, which realize DNNs for training [8] and/or inference [9] in analog circuits. They achieve software-comparable AI performance (e.g., classification accuracy), can run with substantially higher speed and power efficiency compared to digital circuits (e.g., 35 times lower in power consumption [6]), and promise more in the future.

The high efficiency of Analog In-Memory Computing achieved for DNNs is largely due to the efficient implementation of *Vector–Matrix Multiplications*, which are widely used in DNNs, in the crossbar architecture of NVM cells. Vector–matrix multiplication is dominantly the most frequent

Manuscript received 24 November 2023; revised 21 July 2024; accepted 17 August 2024. Date of publication 3 September 2024; date of current version 22 October 2024. This work was supported in part by NSF under Grant CCF-2416361.

The author is with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77845 USA (e-mail: ajiang@cse.tamu.edu).

Communicated by V. Skachek, Associate Editor for Coding and Decoding.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2024.3454059.

Digital Object Identifier 10.1109/TIT.2024.3454059

operation in most DNNs, whether it is a dense network, convolutional network (CNN), recurrent network (RNN), graph network or transformer model. The crossbar architecture is illustrated in Fig. 1 (a). The crossbar array has L row conductors, k column conductors, and Lk nanoscale nonvolatile resistive memories (e.g., memristors [7], phase-change memories [3], [4], etc.) at the junctions. Let $\mathbf{A} = (a_{i,j})_{L\times k}$ be a matrix of non-negative numbers. For $i = 0, 1, \cdots, L-1$ and $j = 0, 1, \cdots, k-1$, the resistor at the junction of the i-th row and the j-th column is programmed to have conductance that is proportional to $a_{i,j}$. Let $\mathbf{u} = (u_0, u_1, \cdots, u_{L-1}) \in \mathbb{R}^L$ be a vector. For $i = 0, 1, \cdots, L-1$, let the input voltage on the i-th row be proportional to u_i. Let $(c_0, c_1, \cdots, c_{k-1}) = \mathbf{uA}$ be the multiplication of the vector $\mathbf{u}$ and the matrix $\mathbf{A}$. Then $(c_0, c_1, \cdots, c_{k-1})$ can be computed by reading the currents at the columns, where for $j = 0, 1, \cdots, k-1$, the current on the j-th column is proportional to c_j. Note that if $\mathbf{A}$ contains negative numbers [6], [10], we can write it as $\mathbf{A} = \mathbf{A}^+ - \mathbf{A}^-$, where $\mathbf{A}^+$ and $\mathbf{A}^-$ are both non-negative matrices, and use two crossbar arrays to compute $(c_0, c_1, \cdots, c_{k-1})$ as $\mathbf{uA}^+ - \mathbf{uA}^-$. In DNNs, the matrix $\mathbf{A}$ represents model parameters (i.e., edge weights in the DNN), which remain constant during inference. The vector $\mathbf{u}$ represents the input to a layer in the DNN, which are variables since their values change for different input samples. Compared to digital computing, which needs Lk scalar multiplications and $(L-1)k$ additions to compute $\mathbf{uA}$, the crossbar can compute $\mathbf{uA}$ in a single time step by exploiting Ohm's law and Kirchhoff's law, thus significantly improving the speed and energy efficiency of computing, potentially by multiple orders.

A challenge for analog in-memory computing, however, is the reliability of computing against errors. Nonvolatile memories are known to have many noise mechanisms, include cell-programming noise, cell-level drifting, random noise, read/write disturbs, stuck cells, short cells, etc. In general, the errors can be partitioned into two types: (1) those that are small but ubiquitous (i.e., appearing in nearly all cells), such as programming noise, cell-level drifting, random noise, etc., and (2) those that are more isolated but can be much more significant, such as stuck cells, short cells (e.g., due to faults in the programming process [10]), memory/circuit defects, *etc*. The two types of errors are modeled by LMEs and UMEs, respectively. DNNs often naturally have some tolerance of small ubiquitous noise [3], [4], [6], [10], [11]. However, they are challenged by significant outlier errors, which need to be detected and corrected.

Analog ECC has been proposed to address the above challenge as follows [1]. Let $\mathcal{C}$ be a linear $[n, k]$ Analog ECC. We extend the $L \times k$ crossbar array for vector–matrix multiplication to an $L \times n$ crossbar array, as illustrated in Fig. 1 (b). Each row in the original matrix $\mathbf{A} = (a_{i,j})_{L\times k}$ is extended to a codeword. That is, for $i = 0, 1, \cdots, L-1$, the i-th row in the matrix, $(a_{i,0}, a_{i,1}, \cdots, a_{i,k-1})$, is encoded into a codeword $\mathbf{c}_i \triangleq (a_{i,0}, a_{i,1}, \cdots, a_{i,n-1})$, and the $n-k$ extra memory cells in the row are programmed so that their conductance values are proportional to $a_{i,k}$, $a_{i,k+1}$, $\cdots$, $a_{i,n-1}$, respectively. By the linearity of the code, no matter what the input variables u_0, u_1, $\cdots$, u_{L-1} are, the output vector $\mathbf{c} \triangleq (c_0, c_1, \cdots, c_{n-1}) = \sum_{i=0}^{L-1} u_i \mathbf{c}_i$ is also a codeword in $\mathcal{C}$. Therefore, significant errors in $\mathbf{c}$ can be corrected by the decoder of $\mathcal{C}$. And note that the first k elements in $\mathbf{c}$ are simply the desired output of the vector–matrix multiplication $\mathbf{uA}$. For more details on the design and experimental performance, please refer to [1], [2], [10], and [12].

Analog ECCs consider LMEs as *tolerable* (as long as δ is small), and focus on the detection and correction of the UMEs, especially those UMEs whose magnitudes exceed the threshold Δ. Given the above considerations, the decoding objective of Analog ECC is set as follows. [1] The decoder for a linear $[n, k]$ Analog ECC $\mathcal{C}$ is a function

$$\mathcal{D} : \mathbb{R}^n \to 2^{[n\rangle}$$

that returns a set of locations of UMEs. Let δ, $\Delta \in R^+$ be positive thresholds with $\delta < \Delta$ as mentioned earlier, and let t be a nonnegative integer. We say that "the decoder $\mathcal{D}$ corrects t UMEs (with respect to the threshold pair (δ, Δ))" if for every possible vector $\mathbf{y} = \mathbf{c}+\boldsymbol{\varepsilon}+\mathbf{e}$ with $\mathbf{c} \in \mathcal{C}$ being a codeword, $\boldsymbol{\varepsilon}$ being an LME vector and $\mathbf{e}$ being an UME vector whose Hamming weight $\mathrm{w}_H(\mathbf{e})$ is at most t, the following condition holds:

$$\mathrm{Supp}_\Delta(\mathbf{e}) \subseteq \mathcal{D}(\mathbf{y}) \subseteq \mathrm{Supp}_0(\mathbf{e}).$$

The above condition not only ensures that the decoder will find all the locations of UMEs whose magnitudes are more than Δ (thus no "false negative"), but also ensures that all the found locations, namely $\mathcal{D}(\mathbf{y})$, have UMEs (thus no "false positive"). [2] After the decoder locates the UMEs (which include as a subset all those *significant* UMEs whose magnitudes exceed Δ), those UMEs can be removed by either re-computing the corresponding entries in the codeword $\mathbf{c}$ (as in the case of the vector-matrix multiplication application where $\mathbf{c}$ is the result of such a multiplication [10], [12]), or by estimating the values of those UMEs via an extended decoding algorithm [1].

In spite of the importance of Analog ECCs for machine learning, the designs of such codes are still relatively limited. Most existing codes focus on the detection or correction of only one UME [1]. A main challenge in the designing of more codes, including codes that correct more than one UME, lies in the analysis of the error-correction capabilities of codes. Such an analysis requires the computing of an important quantity of the code named m-*height* [1], which is analogous to the minimum distance of conventional ECCs over finite fields.

In this paper, we first propose a baseline algorithm that computes the m-height of an Analog ECC. We then present a more efficient algorithm based on linear programming, which reduces the time complexity of computing the m-height by a factor of $(m-1)! \cdot (n-m-1)! \cdot 2^{n-m}$. We use the algorithm to find the exact values of m-heights of existing Analog ECCs for all m. (For many of those codes, only upper bounds to their 1-height or 2-heights were known previously.) The

[1]The original decoding objectives include both error correction and detection. In this work, we focus on error correction alone. So the decoding objective described here is simplified compared to [2].

[2]Note that given a pair of noiseless and noisy codewords $(\mathbf{c}, \mathbf{y})$, there can be different pairs of error vectors $(\boldsymbol{\varepsilon}, \mathbf{e})$ that change $\mathbf{c}$ into $\mathbf{y}$, and the decoding objective needs to be realized for all such possible pairs of error vectors.

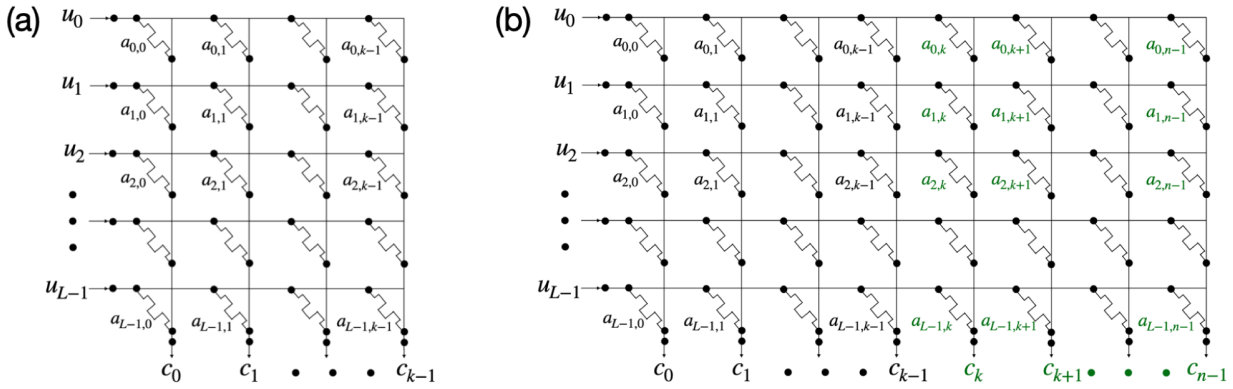

Fig. 1. (a) A crossbar architecture for vector–matrix multiplication. (b) Using Analog ECC for the multiplication, where each row $(a_{i,0}, a_{i,1}, \cdots, a_{i,n-1})$ in the array and the output vector $(c_0, c_1, \cdots, c_{n-1})$ are codewords of the code.

exact m-height values help us understand the error-correction capabilities of the known codes better.

We then present a new family of Analog ECCs called *Analog Permutation Codes*, where the columns in such a code's generator matrix are distinct permutations of the same set of real numbers. We prove that for Analog Permutations Codes, the time complexity of the m-height algorithm can be further reduced by a factor of n. The gained efficiency can help us search for more codes in the code space, and verify their error-correction capabilities.

We then use genetic programming to search for new codes, using randomly generated Analog ECCs and Analog Permutation Codes as seeds. A number of new codes that achieve state-of-the-art performance are discovered, whose m-heights are summarized in Table III. The codes can be used to correct one or more (up to 4) UMEs.

The rest of the paper is organized as follows. In Section II, a survey on existing Analog ECC constructions and their performance is presented, and related works are reviewed. In Section III, an efficient m-height algorithm is presented and analyzed, and the algorithm is used to find the exact m-height values of existing codes. In Section IV, Analog Permutation Codes are analyzed, for which the m-height algorithm's efficiency is further improved. In Section V, new codes that are discovered via genetic programming and achieve state-of-the-art performance are presented. In Section VI, concluding remarks are presented.

II. Existing Constructions for Analog ECCs and Related Works

In this section, we summarize the known constructions for Analog ECCs, with a focus on their error correction – instead of error detection – capabilities. We first review an important analytical tool called m-height and its relation to the error-correction capability of an Analog ECC [1]. Let $\mathbf{x} = (x_0, x_1, \cdots, x_{n-1}) \neq (0, 0, \cdots, 0)$ be a vector in $\mathbb{R}^n$. Let $\pi : [n\rangle \to [n\rangle$ be a permutation such that

$$|x_{\pi(0)}| \geq |x_{\pi(1)}| \geq \cdots \geq |x_{\pi(n-1)}|.$$

For any $m \in [n\rangle$, the m-height of $\mathbf{x}$ is defined as

$$h_m(\mathbf{x}) = \left| \frac{x_{\pi(0)}}{x_{\pi(m)}} \right|$$

if $x_{\pi(m)} \neq 0$, and as $h_m(\mathbf{x}) = \infty$ if $x_{\pi(m)} = 0$. For the all-zero vector $\mathbf{0} = (0, 0, \cdots, 0)$, its m-height is defined as $h_m(\mathbf{0}) = 0$ for all m. Then, the m-height of a linear $[n, k]$ code $\mathcal{C}$ over $\mathbb{R}$ is defined as

$$h_m(\mathcal{C}) = \max_{\mathbf{c} \in \mathcal{C}} h_m(\mathbf{c}).$$

The next important result was proven in [1].

Theorem 1: Let $\mathcal{C}$ be a linear $[n, k]$ code over $\mathbb{R}$. Given δ, $\Delta \in \mathbb{R}^+$ with $\delta < \Delta$ and a positive integer t, there exists a decoder for $\mathcal{C}$ that corrects t UMEs if and only if

$$\Delta \geq 2(h_{2t}(\mathcal{C}) + 1)\delta.$$

We now present the existing constructions for Analog ECCs. Let us start with the *Repetition Code* [1]. Let $\mathcal{C}$ be the $[n, 1]$ repetition code over $\mathbb{R}$, whose generator matrix is the all-one vector $\mathbf{1} = (1, 1, \cdots, 1)$. Its m-height is $h_m(\mathcal{C}) = 1$ for $m \in [n\rangle$. So by Theorem 1, the code can correct $\lfloor (n-1)/2 \rfloor$ UMEs as long as $\Delta \geq 4\delta$.

The next code to consider is the *Cartesian power of repetition code* [1]. Let $\mathcal{C}$ be a linear $[n = wk, k]$ code over $\mathbb{R}$ that is the k-fold Cartesian power of the $[w, 1]$ repetition code. Its generator matrix is a $k \times n$ binary matrix where each row has $n/k = w$ 1s and each column has one 1, while all the remaining elements are 0s. If we use $G_{n=wk,k}$ to denote its generator matrix, then it has the recursive form

$$G_{wk,k} = \left(\begin{array}{cccc|cccc} 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ \hline \multicolumn{4}{c|}{\mathbf{0}} & \multicolumn{4}{c}{G_{w(k-1),k-1}} \end{array} \right).$$

Its m-height is $h_m(\mathcal{C}) = 1$ for $m \in [w\rangle$, and $h_m(\mathcal{C}) = \infty$ for $m \geq w$. So by Theorem 1, the code can correct $\lfloor (w-1)/2 \rfloor$ UMEs as long as $\Delta \geq 4\delta$.

The third code to present has an upper bounded for its 1-height [1]. Although it is not for correcting any UME, it can detect a single UME by the definition of error detection in [2]. Let H be a $r \times n$ binary matrix over $\{0, 1\}$ with $r < n$ that satisfies two properties: (1) every column in H has exactly one 1, and (2) each row of H has either $\lfloor n/r \rfloor$ or $\lceil n/r \rceil$ 1s.

Let $\mathcal{C}$ be a linear $[n, k = n - r]$ code over $\mathbb{R}$ with H as its parity-check matrix. Then its 1-height satisfies

$$h_1(\mathcal{C}) \leq \lceil n/r \rceil - 1.$$

When n is a multiple of r, the code is the dual code of the r-fold Cartesian power of the $[n/r, 1]$ repetition code.

The fourth code to introduce has an upper bound for its 2-height [1], which is useful for correcting one UME. Let r be a positive even integer, and let n be an integer such that $r \leq n \leq r(r-1)$. Let H be a $r \times n$ matrix over $\{-1, 0, 1\}$ that satisfies three properties: (1) all the columns in H are distinct, (2) every column in H has exactly two nonzero entries, the first of which being a 1, and (3) the number of nonzero entries in each row of H is either $\lfloor 2n/r \rfloor$ and $\lceil 2n/r \rceil$. Note that such a matrix H is guaranteed to exist [13]. For example, when $r = 4$ and $n = 12$, H can be [1]:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & -1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 \end{pmatrix}$$

Let $\mathcal{C}$ be a linear $[n, k \geq n - r]$ code over $\mathbb{R}$ with H as its parity-check matrix. Then its 2-height satisfies

$$h_2(\mathcal{C}) \leq \lceil 2n/r \rceil - 1.$$

So by Theorem 1, the code can correct one UME as long as $\Delta/\delta \geq 2\lceil 2n/r \rceil$.

The fifth code is an extension of the fourth code [1]. Recall that the parity-check matrix H of the fourth code satisfies three properties. Now let us generalize the second property as follows: instead of requiring every column of H to have Hamming weight 2, we now require it to have Hamming weight b for some prescribed integer $b \geq 2$. Let $n = \binom{r}{b} \cdot 2^{b-1}$. Then the 2-height of $\mathcal{C}$ satisfies

$$h_2(\mathcal{C}) \leq \lceil bn/r \rceil - 1.$$

So by Theorem 1, the code can correct one UME as long as $\Delta/\delta \geq 2\lceil bn/r \rceil$.

The sixth code to introduce has known finite 2-height values, and therefore is suitable for correcting one UME [1]. Let $n \geq 3$ be an integer, let $\alpha = \pi/n$, and let $\omega = e^{i\alpha}$ with $i = \sqrt{-1}$, namely, ω is the complex primitive $2n$-th root of unity. Let $\mathcal{C}$ be a linear $[n, k = n-2]$ code over $\mathbb{R}$ defined by

$$\mathcal{C} = \left\{ (c_0, c_1, \cdots, c_{n-1}) \in \mathbb{R}^n \ : \ \sum_{j \in [n\rangle} c_j \omega^j = 0 \right\}.$$

$\mathcal{C}$ is a negacyclic code because if $(c_0, c_1, \cdots, c_{n-2}, c_{n-1})$ is a codeword, then so is $(-c_{n-1}, c_0, c_1, \cdots, c_{n-2})$. Its generator polynomial is

$$g(x) = 1 - 2\cos(\alpha)x + x^2,$$

and its parity-check matrix can be $H = (\mathbf{h}_j)_{j \in [n\rangle}$ with

$$\mathbf{h}_j = \begin{pmatrix} \cos(j\alpha) - \cos((j+1)\alpha) \\ \sin(j\alpha) - \sin((j+1)\alpha) \end{pmatrix}.$$

The 2-height of $\mathcal{C}$ satisfies

$$h_2(\mathcal{C}) = \frac{1}{2\sin^2(\pi/(2n))} - 1.$$

So by Theorem 1, the code can correct one UME as long as $\Delta/\delta \geq 1/\sin^2(\pi/(2n))$.

There have been previous works that study the correction of analog noise in different settings. In [14], codes based on chaotic dynamic systems were designed to represent an analog number $x \in \mathbb{R}$ by a sequence of numbers $Y = (y_0, y_1, \cdots, y_{k-1})$, which are transmitted over a channel and received as a sequence of noisy numbers $Z = (z_0, z_1, \cdots, z_{k-1})$, where $z_i = y_i + w_i$ for $i \in [k\rangle$ and w_i is the additive noise. A decoding algorithm recovers the number x approximately as $\hat{x}$ from Z. Assuming that the message x and the noise follow certain distributions (e.g., x is uniformly distributed on the unit interval $[0, 1]$ and w_i's are additive white Gaussian noise), the objective is to minimize the expected distortion $E[(x - \hat{x})^2]$.

The above scheme belongs to a broad field named joint source-channel coding (JSCC), with its original study dating back to Shannon [15] and Kotelnikov [16]. In its more general setting, a sequence of m real numbers $X = (x_0, x_1, \cdots, x_{m-1})$ are encoded as a new sequence $Y = (y_0, y_1, \cdots, y_{k-1})$ and transmitted through a noisy channel. Bandwidth compression or expansion is achieved when $m > k$ or $m < k$, respectively. JSCC is particularly interesting in communications when both the source data and the channel noise have Gaussian distributions. Many solutions are based on space-filling curves, including the well known Archimedean spiral and its variations [15], [17]. It was further shown that instead of parameterized spirals, a functional optimization approach could be used for state-of-the-art performance [18].

Analog coding has also been studied in the area of coded distributed computing. In this area, a commonly studied scenario is that computation over a massive dataset is distributed among a set of worker nodes, and as soon as a sufficiently large subset of worker nodes submit their computation results to a central server, the server can use them to obtain the final result. A coding scheme can be applied to the data and/or computation across the worker nodes, so that the system has resiliency against straggler nodes that can prolong computation, maintains security against malicious nodes that may change the computation results in adversarial ways, and keeps privacy of the dataset despite possible collusion among some worker nodes [19]. In [20], it is shown that analog coding can make the computation more scalable because it avoids quantizing data into a finite field, and can achieve better tradeoffs between privacy and accuracy. In [21], [22], [23], [24], and [25], analog coding for coded distributed computing schemes with numerical stability issues are studied. In the above schemes, if their analog vector-matrix multiplications are performed in analog circuits, Analog ECCs can help correct significant computational errors and make the schemes more robust.

Analog ECC considers both UMEs and LMEs, which makes it bear some similarity with the line of research on analog polynomial recovery in the presence of both outlier errors and inlier errors. In [26], the robust polynomial curve fitting problem is studied where, given the existence of an unknown degree-d polynomial p and n ordered pairs $(x_i, y_i) \in [-1, 1] \times [-1, 1]$ for $i = 1, 2, \cdots, n$ such that $|p(x_i) - y_i| \leq \delta$

for all but ρ fraction of the pairs, the goal is to find a degree-d polynomial q such that $||p-q||_\infty$ is small. The problem is further extended to trigonometric polynomials in [27]. The topic, however, also has clear differences from Analog ECC in its objective and formulations. Its analysis is often probabilistic and needs certain assumptions (e.g., the sparsity of outlier errors in every region), while the analysis of Analog ECCs usually focuses on their worst-case performance.

Another related field is ECCs for nonvolatile memories (NVMs). (Note that when vector-matrix multiplications – an important application of Analog ECCs – are implemented in nanoscale analog circuits, the numbers in the matrix are stored in crossbar arrays of NVM cells [3], [4], [6], [10], [12].) Each NVM cell has q discrete levels (e.g., $q = 2$, 4, 8, 16 or 32) and can store $\log_2 q$ bits. ECCs have been designed to correct bit errors [28], limited-magnitude errors [29], [30], [31], errors with graded magnitude distributions [32] and stuck-cell errors [33]. There are also ECCs that inherently use features of NVMs, such as the asymmetric costs of writing data and erasing data in NAND Flash Memories [34], [35], [36], [37]. A variety of techniques have been used to correct errors of different metrics. For instance, to correct Hamming errors in the binary pages of NVMs (sometimes with additional soft information on the sizes of errors), BCH codes and LDPC codes are optimized for NVM channels [28]. To correct asymmetric limited-magnitude errors, a modular method is used to map large-alphabet codes to small-alphabet codes [29]. To correct errors measured by the L_1-metric or Lee-metric in q-ary NVM cells, codes based on multisets, elementary symmetric functions and additive number theory are designed [38], [39], [40], [41]. To correct inter-cell interference (ICI) errors, constrained coding techniques that enable page separation are designed [42]. To correct errors measured by the Kendall's τ-distance in the rank modulation scheme (RM), metric-embedding techniques are used to translate the codes to Lee-metric codes [35], [43]. To correct errors measured by the Ulam metric in RM, new code-interleaving techniques are developed [44]. To correct errors measured by the Infinity Norm in RM, code construction methods based on the direct/semi-direct product of (subgroups of) symmetric groups are used [45]. Such techniques can potentially be useful for the designs of Analog ECCs, too.

Analog ECC differs from the JSCC paradigm in that it does not depend on the probabilistic distributions of data and noise (namely, it optimizes the worst-case performance), differs from the ECC-for-NVM paradigm in that it focuses on analog values (instead of discrete values) for both data and errors, and differs from both paradigms in that it considers two types of errors LME and UME (instead of only one). By tolerating small LMEs and combatting large UMEs, it aims at making machine learning algorithms (especially deep neural networks) run more reliably in next-generation analog computers.

III. Finding the m-Height of Analog ECC

The m-height of Analog ECC is analogous to the minimum distance of conventional error-correcting codes (e.g., codes over finite fields), as evidenced by Theorem 1. It is crucial for finding the error-correction capability of a code.

The Minimum Distance Problem for linear error-correcting codes over finite fields – namely, given a generator matrix $G \in \mathbb{F}_q^{k\times n}$, find the minimum distance d of the corresponding code – is a fundamental computational problem in coding theory [46]. However, the problem is computationally intractable. In [47], Vardy proved that the minimum distance cannot be computed exactly in deterministic polynomial time unless P = NP. In [46], Dumer, Micciancio and Sudan proved that the minimum distance is not approximable to within any constant factor in random polynomial time (RP) unless NP = RP. They also showed that the minimum distance is not approximable to within an additive error that is linear in the block length n of the code. It was further proven that under the stronger assumption that NP is not contained in RQP (random quasi-polynomial time), the minimum distance is not approximable to within the factor $2^{\log^{1-\epsilon}(n)}$ for any $\epsilon > 0$. The hardness of the Minimum Distance Problem has also been shown for error-correcting codes in more domains. For instance, for quantum codes, Kapshikar and Kundu have proved that it is NP-hard to find the minimum distance of stabilizer quantum codes either exactly or approximately [48].

For Analog ECC, the corresponding m-Height Problem can be defined as follows: *given a generator matrix $G \in \mathbb{R}^{k\times n}$, find the m-height of the corresponding Analog ECC, where* $m \in [n\rangle$. Note that the m-Height Problem is more general than finding the minimum distance $d(\mathcal{C})$ of the Analog ECC $\mathcal{C}$. Let $\mathrm{w}_H(\mathbf{c})$ denote the Hamming weight of a codeword $\mathbf{c}$. Then $d(\mathcal{C})$ equals the minimum Hamming weight of a nonzero codeword in $\mathcal{C}$, namely, $d(\mathcal{C}) = \min_{\mathbf{c}\in\mathcal{C}-\{\mathbf{0}\}} \mathrm{w}_H(\mathbf{c})$. Based on the definition of m-height, we have

$$h_0(\mathcal{C}) \le h_1(\mathcal{C}) \le \cdots \le h_{n-1}(\mathcal{C}).$$

Let $h_n(\mathcal{C}) \triangleq \infty$. Then $d(\mathcal{C})$ is the minimum index $m \in [n+1\rangle$ such that $h_m(\mathcal{C}) = \infty$. Namely, $h_m(\mathcal{C}) = \infty$ if and only if $m \ge d(\mathcal{C})$. So when the m-height values are found for all $m \in [n\rangle$, the value of $d(\mathcal{C})$ also becomes known. Knowing the m-height values is also more important for analyzing the error-correction capability of an Analog ECC than simply knowing $d(\mathcal{C})$, because the necessary and sufficient condition for the code to be able to correct t UMEs is $\Delta/\delta \ge 2(h_{2t}(\mathcal{C})+1)$, which depends on the ratio Δ/δ. The value of $d(\mathcal{C})$ can tell us that the code can potentially correct $\lfloor \frac{1}{2}(d(\mathcal{C})-1)\rfloor$ UMEs, however it cannot guarantee that any $\lfloor \frac{1}{2}(d(\mathcal{C})-1)\rfloor$ UMEs are correctable without knowing the ratio Δ/δ; that is, $d(\mathcal{C})$ provides only a necessary but not sufficient condition for the error correction capability.

In this section, we study how to find the m-height of an Analog ECC given its generator matrix. We first present a baseline algorithm, which solves the m-Height Problem by solving $n! \cdot 2^n$ linear-fractional programs. We then improve the computational complexity substantially by presenting an enhanced algorithm that discovers the m-height via solving $n(n-1)\binom{n-2}{m-1}2^m$ linear programs, which reduces the complexity by a factor of $(m-1)!\cdot(n-m-1)!\cdot 2^{n-m}$ compared to the baseline method.

A. A Baseline Algorithm for m-Height Problem

Consider a linear $[n, k]$ code $\mathcal{C}$ over $\mathbb{R}$. For any non-empty subset of codewords $\tilde{C} \subseteq \mathcal{C}$, we define

$$h_m(\tilde{C}) = \max_{\mathbf{c}\in\tilde{C}} h_m(\mathbf{c}).$$

And we define $h_m(\emptyset) = 0$.

Let $\pi : [n\rangle \to [n\rangle$ denote a permutation on $[n\rangle$, and let Π denote the set of all the $n!$ such permutations. Let $\mathbf{s} = (s_0, s_1, \cdots, s_{n-1}) \in \{1, -1\}^n$ be a binary vector of length n, and let $\mathbf{S} = \{1, -1\}^n$ denote the set of all the 2^n such vectors. Let sgn be the sign function:

$$\text{sgn}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Let $\mathcal{C}_{\pi,\mathbf{s}}$ denote the subset of codewords of $\mathcal{C}$ such that a codeword $\mathbf{c} = (c_0, c_1, \cdots, c_n) \in \mathcal{C}$ is in $\mathcal{C}_{\pi,\mathbf{s}}$ if $\mathbf{c} \neq \mathbf{0}$, $\text{sgn}(c_{\pi(j)}) = s_j$ for $j \in [n\rangle$, and

$$|c_{\pi(j)}| = s_j \cdot c_{\pi(j)} \;\geq\; s_{j+1} \cdot c_{\pi(j+1)} = |c_{\pi(j+1)}|$$

for $j \in [n-1\rangle$. Since $\mathcal{C} - \{\mathbf{0}\} = \bigcup_{\pi\in\Pi,\mathbf{s}\in\mathbf{S}} \mathcal{C}_{\pi,\mathbf{s}}$ and $h_m(\mathbf{0}) = 0$, we get

$$h_m(\mathcal{C}) = \max_{\pi\in\Pi,\mathbf{s}\in\mathbf{S}} h_m(\mathcal{C}_{\pi,\mathbf{s}}).$$

Lemma 1: Let $\mathcal{C}$ be a linear $[n, k]$ code over $\mathbb{R}$. Let $G = (g_{i,j})_{k\times n} \in \mathbb{R}^{k\times n}$ be the generator matrix of $\mathcal{C}$. Let $m \in [d(\mathcal{C})\rangle - \{0\}$, $\pi \in \Pi$ and $\mathbf{s} = (s_0, s_1, \cdots, s_{n-1}) \in \mathbf{S}$. Let $F_{\pi,\mathbf{s}}$ denote the following linear-fractional program with k real-valued variables u_0, u_1, $\cdots$, u_{k-1}:

$$\begin{aligned} \text{maximize} \quad & \frac{s_0 \cdot \sum_{i\in[k\rangle} \left(u_i g_{i,\pi(0)}\right)}{s_m \cdot \sum_{i\in[k\rangle} \left(u_i g_{i,\pi(m)}\right)} \\ \text{s.t.} \quad & s_0 \cdot \sum_{i\in[k\rangle} \left(u_i g_{i,\pi(0)}\right) > 0 \\ & s_{n-1} \cdot \sum\nolimits_{i\in[k\rangle} \left(u_i g_{i,\pi(n-1)}\right) \geq 0 \\ & s_j \cdot \sum\nolimits_{i\in[k\rangle} \left(u_i g_{i,\pi(j)}\right) \geq s_{j+1} \cdot \sum\nolimits_{i\in[k\rangle} \left(u_i g_{i,\pi(j+1)}\right) \\ & \quad \forall j \in [n-1\rangle \end{aligned}$$

Let $f_{\pi,\mathbf{s}}$ be the optimal objective value of $F_{\pi,\mathbf{s}}$ if $F_{\pi,\mathbf{s}}$ is feasible, and let $f_{\pi,\mathbf{s}} = 0$ otherwise. Then

$$h_m(\mathcal{C}) = \max_{\pi\in\Pi,\mathbf{s}\in\mathbf{S}} f_{\pi,\mathbf{s}}$$

Proof: A vector $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1})$ is a codeword in $\mathcal{C}$ if and only if there is a vector $\mathbf{u} = (u_0, u_1, \cdots, u_{k-1}) \in \mathbb{R}^k$ such that $\mathbf{c} = \mathbf{u}G$. The constraints in $F_{\pi,\mathbf{s}}$ provide the necessary and sufficient condition for $\mathbf{u}G$ to be a codeword in $\mathcal{C}_{\pi,\mathbf{s}}$, and the objective function of $F_{\pi,\mathbf{s}}$ is the m-height of $\mathbf{u}G$. (The condition $s_0 \cdot \sum_{i\in[k\rangle} \left(u_i g_{i,\pi(0)}\right) > 0$ ensures that the codeword is not an all-zero codeword. And since $\sum_{i\in[k\rangle} \left(u_i g_{i,\pi(m)}\right)$ is the number in the codeword $\mathbf{u}G$ with the $(m+1)$-th highest absolute value, it cannot be 0 when $m < d(\mathcal{C})$, so the objective function of $F_{\pi,\mathbf{s}}$ is well defined.) Therefore $f_{\pi,\mathbf{s}} = h_m(\mathcal{C}_{\pi,\mathbf{s}})$. Since $h_m(\mathcal{C}) = \max_{\pi\in\Pi,\mathbf{s}\in\mathbf{S}} h_m(\mathcal{C}_{\pi,\mathbf{s}})$, we get $h_m(\mathcal{C}) = \max_{\pi\in\Pi,\mathbf{s}\in\mathbf{S}} f_{\pi,\mathbf{s}}$. □

The lemma above considers $h_m(\mathcal{C})$ for $m > 0$ because $h_0(\mathcal{C}) \equiv 1$ as long as $\mathcal{C}$ contains a nonzero codeword. It indicates a baseline method for finding the m-height of an Analog Code: solve $n! \cdot 2^n$ linear-fractional programs $F_{\pi,\mathbf{s}}$, and take the maximum of their corresponding values of $f_{\pi,\mathbf{s}}$. It turns the m-Height Problem to a computational problem. But when n is large, the number of linear-fractional programs to solve, $n! \cdot 2^n$, is still prohibitively high. In the following, we present an improved method that makes the computation substantially more efficient.

B. More Efficient Algorithm for m-Height Problem

Let $m \in [n\rangle - \{0\}$. Let $\Psi = \{-1, 1\}^m$ be the set of 2^m binary vectors of length m whose elements are either 1 or -1.

Let (a, b, X, ψ) be a tuple where $a \in [n\rangle$, $b \in [n\rangle - \{a\}$, $X \subseteq [n\rangle - \{a, b\}$, $|X| = m-1$, and $\psi = (s_0, s_1, \cdots, s_{m-1}) \in \Psi$. Let Γ denote the set of all the

$$n(n-1)\binom{n-2}{m-1}2^m$$

such tuples.

Given a tuple $(a, b, X, \psi) \in \Gamma$, let $x_1, x_2, \cdots, x_{m-1}$ denote the $m-1$ integers in X such that

$$x_1 < x_2 < \cdots < x_{m-1}.$$

Define $Y \triangleq [n\rangle - X - \{a, b\}$, and let $x_{m+1}, x_{m+2}, \cdots, x_{n-1}$ denote the $n-m-1$ integers in Y such that

$$x_{m+1} < x_{m+2} < \cdots < x_{n-1}.$$

Let $x_0 = a$ and $x_m = b$. Then $x_0, x_1, \cdots, x_{n-1}$ are the n distinct integers in $[n\rangle$. Let τ denote the permutation on $[n\rangle$ such that $\tau(j) = x_j$ for $j \in [n\rangle$. We call τ the *quasi-sorted permutation given* (a, b, X, ψ). Let $\mathcal{C}_{a,b,X,\psi}$ denote a subset of nonzero codewords of $\mathcal{C}$ such that a nonzero codeword $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in \mathcal{C}$ is in $\mathcal{C}_{a,b,X,\psi}$ if and only if it satisfies the following properties:

1) For $j = 1, 2, \cdots, m-1$, $|c_{\tau(0)}| \geq |c_{\tau(j)}| \geq |c_{\tau(m)}|$.
2) For $j = m+1, m+2, \cdots, n-1$, $|c_{\tau(m)}| \geq |c_{\tau(j)}|$.
3) $\forall\ j \in [m\rangle$, $\text{sgn}(c_{\tau(j)}) = s_j$. (Note that here s_j is the j-th element of ψ.)

For any nonzero codeword $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in \mathcal{C}$, there exists at least one tuple $(a, b, X, \psi) \in \Gamma$ such that $\mathbf{c} \in \mathcal{C}_{a,b,X,\psi}$. (To see that, let $\pi \in \Pi$ be a permutation such that $|c_{\pi(0)}| \geq |c_{\pi(1)}| \geq \cdots \geq |c_{\pi(n-1)}|$. Then we let $a = \pi(0)$, $b = \pi(m)$, $X = \{\pi(1), \pi(2), \cdots, \pi(m-1)\}$. Let $x_1, x_2, \cdots, x_{m-1}$ denote the $m-1$ integers in X such that $x_1 < x_2 < \cdots < x_{m-1}$, and let $x_0 = a$. Then we let $\psi = (s_0, s_1, \cdots, s_{m-1})$ where $\forall\ j \in [m\rangle$, $s_j = \text{sgn}(c_{x_j})$. For the above tuple (a, b, X, ψ), we have $\mathbf{c} \in \mathcal{C}_{a,b,X,\psi}$.) Therefore we have $\mathcal{C} - \{\mathbf{0}\} = \bigcup_{(a,b,X,\psi)\in\Gamma} \mathcal{C}_{a,b,X,\psi}$. Since $h_m(\mathbf{0}) = 0$, we get

$$h_m(\mathcal{C}) = \max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{C}_{a,b,X,\psi})$$

Theorem 2: Let $\mathcal{C}$ be a linear $[n, k]$ code over $\mathbb{R}$. Let $G = (g_{i,j})_{k\times n} \in \mathbb{R}^{k\times n}$ be a generator matrix of $\mathcal{C}$ where no column is $\mathbf{0}$. Let $d(\mathcal{C})$ be the minimum distance of $\mathcal{C}$, and let $m \in \{1, 2, \cdots, \min\{d(\mathcal{C}), n-1\}\}$. Let $(a, b, X, \psi) \in \Gamma$,

where $\psi = (s_0, s_1, \cdots, s_{m-1})$. Define $Y \triangleq [n\rangle - X - \{a, b\}$, and let τ be the quasi-sorted permutation given (a, b, X, ψ). Let $LP_{a,b,X,\psi}$ denote the following linear program with k real-valued variables u_0, u_1, $\cdots$, u_{k-1}:

$$\begin{aligned}
\text{maximize } & \textstyle\sum_{i\in[k\rangle} (s_0 g_{i,a}) \cdot u_i \\
\text{s.t. } & \textstyle\sum_{i\in[k\rangle} (s_{\tau^{-1}(j)} g_{i,j} - s_0 g_{i,a}) \cdot u_i \le 0 \quad \text{for } j \in X \\
& \textstyle\sum_{i\in[k\rangle} (-s_{\tau^{-1}(j)} g_{i,j}) \cdot u_i \le -1 \quad \text{for } j \in X \\
& \textstyle\sum_{i\in[k\rangle} g_{i,b} \cdot u_i = 1 \\
& \textstyle\sum_{i\in[k\rangle} g_{i,j} \cdot u_i \le 1 \quad \text{for } j \in Y \\
& \textstyle\sum_{i\in[k\rangle} -g_{i,j} \cdot u_i \le 1 \quad \text{for } j \in Y
\end{aligned}$$

Let $z_{a,b,X,\psi}$ be the optimal objective value of $LP_{a,b,X,\psi}$ if it is bounded, let $z_{a,b,X,\psi} = \infty$ if the optimal objective value of $LP_{a,b,X,\psi}$ is unbounded, and let $z_{a,b,X,\psi} = 0$ if $LP_{a,b,X,\psi}$ is infeasible. Then

$$h_m(\mathcal{C}) = \max_{(a,b,X,\psi)\in\Gamma} z_{a,b,X,\psi}$$

Proof: Note that $h_m(\mathcal{C}) = \infty$ if and only if $m \ge d(\mathcal{C})$. Let us first consider the case $m < d(\mathcal{C})$. In this case, given any tuple $(a, b, X, \psi) \in \Gamma$, let $\mathcal{Z}_{a,b,X,\psi}$ denote the subset of codewords of $\mathcal{C}_{a,b,X,\psi}$ such that a codeword $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in \mathcal{C}_{a,b,X,\psi}$ is in $\mathcal{Z}_{a,b,X,\psi}$ if and only if $c_b = 1$. Since $\mathcal{Z}_{a,b,X,\psi} \subseteq \mathcal{C}_{a,b,X,\psi}$, we have $h_m(\mathcal{Z}_{a,b,X,\psi}) \le h_m(\mathcal{C}_{a,b,X,\psi})$. Since $h_m(\mathcal{C}) = \max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{C}_{a,b,X,\psi})$, we get $h_m(\mathcal{C}) \ge \max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{Z}_{a,b,X,\psi})$.

On the other side, for any codeword $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in \mathcal{C}_{a,b,X,\psi}$, consider the vector $\hat{\mathbf{c}} = (\hat{c}_0, \hat{c}_1, \cdots, \hat{c}_{n-1}) \triangleq (1/c_b) \cdot \mathbf{c}$. Since $\mathbf{c} \in \mathcal{C}_{a,b,X,\psi}$, there exists a quasi-sorted permutation τ on $[n\rangle$ such that for any $j \in \{1, 2, \cdots, m-1\}$ and $l \in \{m+1, m+2, \cdots, n-1\}$, we have $|c_{\tau(0)}| \ge |c_{\tau(j)}| \ge |c_{\tau(m)}| = |c_b| \ge |c_{\tau(l)}|$. Therefore, if we sort the n elements in $\mathbf{c}$ by their absolute values from the highest to the lowest, c_b will be the $(m+1)$-th element. Since $\mathbf{c} \ne \mathbf{0}$ and $m + 1 \le d(\mathcal{C})$, as every nonzero codeword has at least $d(\mathcal{C})$ nonzero elements, we have $c_b \ne 0$, which means the vector $\hat{\mathbf{c}}$ is well defined.

Let $\mathbf{u} \in \mathbb{R}^k$ be the vector such that $\mathbf{c} = \mathbf{u}G$. Then $\hat{\mathbf{u}} \triangleq (1/c_b) \cdot \mathbf{u}$ is the vector such that $\hat{\mathbf{c}} = \hat{\mathbf{u}}G$. So $\hat{\mathbf{c}} \in \mathcal{C}$. Since $\hat{\mathbf{c}}$ is just a scaled version of $\mathbf{c}$ – namely, $\forall\ j \in [n\rangle$, $\hat{c}_j = c_j/c_b$ – and $\mathbf{c} \in \mathcal{C}_{a,b,X,\psi}$, the codeword $\hat{\mathbf{c}}$ also satisfies the first two properties of $\mathcal{C}_{a,b,X,\psi}$, that is: (1) for $j = 1, 2, \cdots, m-1$, $|\hat{c}_{\tau(0)}| \ge |\hat{c}_{\tau(j)}| \ge |\hat{c}_{\tau(m)}|$, and (2) for $j = m+1, m+2, \cdots, n-1$, $|\hat{c}_{\tau(m)}| \ge |\hat{c}_{\tau(j)}|$. As to the third property of $\mathcal{C}_{a,b,X,\psi}$, if $c_b > 0$, we have $\text{sgn}(\hat{c}_{\tau(j)}) = \text{sgn}(c_{\tau(j)}/c_b) = \text{sgn}(c_{\tau(j)}) = s_j$ for $j \in [m\rangle$, which leads to $\hat{\mathbf{c}} \in \mathcal{C}_{a,b,X,\psi}$; otherwise $c_b < 0$, and we have $\text{sgn}(\hat{c}_{\tau(j)}) = \text{sgn}(c_{\tau(j)}/c_b) = -\text{sgn}(c_{\tau(j)}) = -s_j$ for $j \in [m\rangle$, which leads to $\hat{\mathbf{c}} \in \mathcal{C}_{a,b,X,-\psi}$, where $-\psi = -(s_0, s_1, \cdots, s_{m-1}) = (-s_0, -s_1, \cdots, -s_{m-1})$. Therefore $\hat{\mathbf{c}} \in \mathcal{C}_{a,b,X,\psi} \cup \mathcal{C}_{a,b,X,-\psi}$ in any case. Since $\hat{c}_b = c_b/c_b = 1$, we get $\hat{\mathbf{c}} \in \mathcal{Z}_{a,b,X,\psi} \cup \mathcal{Z}_{a,b,X,-\psi}$. Since $h_m(\mathbf{c}) = |c_{\tau(0)}|/|c_{\tau(m)}| = |c_{\tau(0)}/c_b|/|c_{\tau(m)}/c_b| = |\hat{c}_{\tau(0)}|/|\hat{c}_{\tau(m)}| = h_m(\hat{\mathbf{c}})$, and $\mathbf{c}$ can be any codeword in $\mathcal{C}_{a,b,X,\psi}$, we get

$$h_m(\mathcal{C}_{a,b,X,\psi}) \le \max\{h_m(\mathcal{Z}_{a,b,X,\psi}), h_m(\mathcal{Z}_{a,b,X,-\psi})\}.$$

Since $h_m(\mathcal{C}) = \max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{C}_{a,b,X,\psi})$, we get $h_m(\mathcal{C})$

$$\begin{aligned}
&\le \max_{(a,b,X,\psi)\in\Gamma} \max\{h_m(\mathcal{Z}_{a,b,X,\psi}), h_m(\mathcal{Z}_{a,b,X,-\psi})\} \\
&= \max\{\max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{Z}_{a,b,X,\psi}), \\
&\qquad \max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{Z}_{a,b,X,-\psi})\} \\
&= \max\{\max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{Z}_{a,b,X,\psi}), \\
&\qquad \max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{Z}_{a,b,X,\psi})\} \\
&= \max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{Z}_{a,b,X,\psi})
\end{aligned}$$

Since we also have $h_m(\mathcal{C}) \ge \max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{Z}_{a,b,X,\psi})$, we get

$$h_m(\mathcal{C}) = \max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{Z}_{a,b,X,\psi})$$

Let us now consider $h_m(\mathcal{Z}_{a,b,X,\psi})$, which equals

$$\begin{aligned}
&\max_{\mathbf{c}=(c_0,c_1,\cdots,c_{n-1})\in\mathcal{Z}_{a,b,X,\psi}} h_m(\mathbf{c}) \\
&= \max_{\mathbf{c}\in\mathcal{Z}_{a,b,X,\psi}} |c_{\tau(0)}/c_{\tau(m)}| \\
&= \max_{\mathbf{c}\in\mathcal{Z}_{a,b,X,\psi}} |c_{\tau(0)}/c_b| \\
&= \max_{\mathbf{c}\in\mathcal{Z}_{a,b,X,\psi}} |c_{\tau(0)}/1| \\
&= \max_{\mathbf{c}\in\mathcal{Z}_{a,b,X,\psi}} |c_{\tau(0)}|
\end{aligned}$$

if $\mathcal{Z}_{a,b,X,\psi} \ne \emptyset$, and equals 0 otherwise. Let us first consider the sub-case where $\mathcal{Z}_{a,b,X,\psi} \ne \emptyset$. Every codeword $\mathbf{c} \in \mathcal{Z}_{a,b,X,\psi}$ is equal to $\mathbf{u}G$ for some vector $\mathbf{u} = (u_0, u_1, \cdots, u_{k-1}) \in \mathbb{R}^k$, and it needs to satisfy the three properties of $\mathcal{C}_{a,b,X,\psi}$ plus the property that $c_{\tau(m)} = 1$ (since $\tau(m) = b$). Since $G = (g_{i,j})_{k\times n}$, $\forall\ j \in [n\rangle$, $c_{\tau(j)} = \sum_{i\in[k\rangle} u_i g_{i,\tau(j)}$. So the value of $h_m(\mathcal{Z}_{a,b,X,\psi})$, which equals

$$\max_{\mathbf{u}\in\mathbb{R}^k, \mathbf{u}G\in\mathcal{Z}_{a,b,X,\psi}} \left| \sum_{i\in[k\rangle} u_i g_{i,\tau(0)} \right|,$$

is the optimal objective value of the following problem:

$$\begin{aligned}
\text{maximize } & |\textstyle\sum_{i\in[k\rangle} u_i g_{i,\tau(0)}| \\
\text{s.t. } & |\textstyle\sum_{i\in[k\rangle} u_i g_{i,\tau(0)}| \ge |\sum_{i\in[k\rangle} u_i g_{i,\tau(j)}| \\
& \qquad \text{for } j = 1, 2, \cdots, m-1 \\
& |\textstyle\sum_{i\in[k\rangle} u_i g_{i,\tau(j)}| \ge |\sum_{i\in[k\rangle} u_i g_{i,\tau(m)}| \\
& \qquad \text{for } j = 1, 2, \cdots, m-1 \\
& \textstyle\sum_{i\in[k\rangle} u_i g_{i,\tau(m)} = 1 \\
& |\textstyle\sum_{i\in[k\rangle} u_i g_{i,\tau(m)}| \ge |\sum_{i\in[k\rangle} u_i g_{i,\tau(j)}| \\
& \qquad \text{for } j = m+1, m+2, \cdots, n-1 \\
& \text{sgn}(\textstyle\sum_{i\in[k\rangle} u_i g_{i,\tau(j)}) = s_j \\
& \qquad \text{for } j = 0, 1, \cdots, m-1
\end{aligned}$$

Note that $\tau(0) = a$, $\tau(m) = b$, and $|\sum_{i\in[k\rangle} u_i g_{i,\tau(j)}| = \text{sgn}(\sum_{i\in[k\rangle} u_i g_{i,\tau(j)}) \cdot (\sum_{i\in[k\rangle} u_i g_{i,\tau(j)})$ for all $j \in [n\rangle$. So for $j \in [m\rangle$, $|\sum_{i\in[k\rangle} u_i g_{i,\tau(j)}| = s_j \sum_{i\in[k\rangle} u_i g_{i,\tau(j)} = \sum_{i\in[k\rangle} (s_j g_{i,\tau(j)}) \cdot u_i$. The permutation τ imposes a one-to-one mapping from $\{1, 2, \cdots, m-1\}$ to the $m-1$ indices

in X, therefore each $s_j g_{i,\tau(j)}$ for $j \in \{1,2,\cdots,m-1\}$ can be expressed as $s_{\tau^{-1}(j')} g_{i,j'}$ for $j' \triangleq \tau(j) \in X$. The permutation τ also imposes a one-to-one mapping from $\{m+1, m+2, \cdots, n-1\}$ to Y, therefore the condition "$|\sum_{i\in[k\rangle} u_i g_{i,\tau(j)}| \leq |\sum_{i\in[k\rangle} u_i g_{i,\tau(m)}| = 1$ for $j = m+1, m+2, \cdots, n-1$" can be expressed as "$-1 \leq \sum_{i\in[k\rangle} u_i g_{i,j} \leq 1$ for $j \in Y$". Consequently, it can be seen that the above optimization problem is equivalent to $LP_{a,b,X,\psi}$, and therefore

$$h_m(Z_{a,b,X,\psi}) = z_{a,b,X,\psi}$$

when $\mathcal{Z}_{a,b,X,\psi} \neq \emptyset$. When $\mathcal{Z}_{a,b,X,\psi} = \emptyset$, we will have $h_m(\mathcal{Z}_{a,b,X,\psi}) = 0$ by definition, and $LP_{a,b,X,\psi}$ will be infeasible, which will mean $z_{a,b,X,\psi} = 0$. So $h_m(Z_{a,b,X,\psi}) = z_{a,b,X,\psi}$ in any case. Since $h_m(\mathcal{C}) = \max_{(a,b,X,\psi)\in\Gamma} h_m(\mathcal{Z}_{a,b,X,\psi})$, we get

$$h_m(\mathcal{C}) = \max_{(a,b,X,\psi)\in\Gamma} z_{a,b,X,\psi}$$

when $m < d(\mathcal{C})$. So the theorem holds when $m < d(\mathcal{C})$.

Now consider the case where $m = d(\mathcal{C})$. In this case, $h_m(\mathcal{C}) = \infty$, and there exists a nonzero codeword $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in \mathcal{C}$ of Hamming weight $d(\mathcal{C})$ and a permutation $\pi \in \Pi$ such that $|c_{\pi(0)}| \geq |c_{\pi(1)}| \geq \cdots \geq |c_{\pi(m-1)}| > |c_{\pi(m)}| = |c_{\pi(m+1)}| = \cdots = |c_{\pi(n-1)}| = 0$. Let $\mathbf{u} \in \mathbb{R}^k$ be the length-k vector such that $\mathbf{c} = \mathbf{u}G$. Let $(i^*, b^*) \in \{0,1,\cdots,k-1\} \times \{\pi(m), \pi(m+1), \cdots, \pi(n-1)\}$ be the pair of indexes such that

$$|g_{i^*,b^*}| = \max_{i\in\{0,1,\cdots,k-1\}, j\in\{\pi(m),\pi(m+1),\cdots,\pi(n-1)\}} |g_{i,j}|$$

Since no column of the generator matrix $G = (g_{i,j})_{k\times n}$ is an all-zero vector, we have $g_{i^*,b^*} \neq 0$. Let $\epsilon \geq 0$ be a non-negative real number, whose value will be discussed below. Let $\mathbf{u}^\epsilon \triangleq (u_0, \cdots, u_{i^*-1}, u_{i^*} + \epsilon, u_{i^*+1}, \cdots, u_{k-1})$, and let $\mathbf{c}^\epsilon = (c_0^\epsilon, c_1^\epsilon, \cdots, c_{n-1}^\epsilon) \triangleq \mathbf{u}^\epsilon G = \mathbf{u}G + (0, \cdots, 0, \epsilon, 0, \cdots, 0)G = \mathbf{c} + \epsilon(g_{i^*,0}, g_{i^*,1}, \cdots, g_{i^*,n-1}) = (c_0 + \epsilon g_{i^*,0}, c_1 + \epsilon g_{i^*,1}, \cdots, c_{n-1} + \epsilon g_{i^*,n-1})$. Let v be the greatest integer in $[m\rangle$ such that $|c_{\pi(0)}| = |c_{\pi(1)}| = \cdots = |c_{\pi(v)}|$. There exists a sufficiently small number $\epsilon_0 > 0$ such that whenever $0 \leq \epsilon \leq \epsilon_0$, all the following conditions are satisfied:

- Condition 1: $\forall\ j \in [m\rangle$, $|c^\epsilon_{\pi(j)}| > |c^\epsilon_{b^*}|$. To satisfy this condition, it is sufficient to have

  $$\epsilon_0 < \min_{j\in[m\rangle} \frac{|c_{\pi(j)}|}{|g_{i^*,\pi(j)}| + |g_{i^*,b^*}|},$$

  because then for all $j \in [m\rangle$, we have $|c^\epsilon_{\pi(j)}| = |c_{\pi(j)} + \epsilon g_{i^*,\pi(j)}| \geq |c_{\pi(j)}| - \epsilon|g_{i^*,\pi(j)}| = |c_{\pi(j)}| - \epsilon(|g_{i^*,\pi(j)}| + |g_{i^*,b^*}|) + \epsilon|g_{i^*,b^*}| > \epsilon|g_{i^*,b^*}| = |0 + \epsilon g_{i^*,b^*}| = |c_{b^*} + \epsilon g_{i^*,b^*}| = |c^\epsilon_{b^*}|$.
  Since $\forall\ j \in \{m, m+1, \cdots, n-1\}$, we have $|g_{i^*,b^*}| \geq |g_{i^*,\pi(j)}|$ and $c^\epsilon_{\pi(j)} = c_{\pi(j)} + \epsilon g_{i^*,\pi(j)} = 0 + \epsilon g_{i^*,\pi(j)} = \epsilon g_{i^*,\pi(j)}$, we get $|c^\epsilon_{b^*}| = \epsilon|g_{i^*,b^*}| \geq \epsilon|g_{i^*,\pi(j)}| = |c^\epsilon_{\pi(j)}|$. So Condition 1 implies that

  $$\min_{j\in[m\rangle} |c^\epsilon_{\pi(j)}| > |c^\epsilon_{b^*}| \geq \max_{j\in\{m,m+1,\cdots,n-1\}-\{\pi^{-1}(b^*)\}} |c^\epsilon_{\pi(j)}|$$

- Condition 2: $\forall\ j \in [m\rangle$, $\mathrm{sgn}(c^\epsilon_{\pi(j)}) = \mathrm{sgn}(c_{\pi(j)})$. To satisfy this condition, it is sufficient to have

  $$\epsilon_0 < \min_{j\in[m\rangle, g_{i^*,\pi(j)}\neq 0} \frac{|c_{\pi(j)}|}{|g_{i^*,\pi(j)}|}$$

  if $\{j \in [m\rangle | g_{i^*,\pi(j)} \neq 0\} \neq \emptyset$ (and no constraint for ϵ_0 otherwise) because then for all $j \in [m\rangle$, if $\mathrm{sgn}(c_{\pi(j)}) = 1$, which means $c_{\pi(j)} \geq 0$, we get $c^\epsilon_{\pi(j)} = c_{\pi(j)} + \epsilon g_{i^*,\pi(j)} \geq |c_{\pi(j)}| - \epsilon|g_{i^*,\pi(j)}| \geq 0$, so $\mathrm{sgn}(c^\epsilon_{\pi(j)}) = 1 = \mathrm{sgn}(c_{\pi(j)})$; otherwise $\mathrm{sgn}(c_{\pi(j)}) = -1$, which means $c_{\pi(j)} < 0$, we get $c^\epsilon_{\pi(j)} = c_{\pi(j)} + \epsilon g_{i^*,\pi(j)} \leq -|c_{\pi(j)}| + \epsilon|g_{i^*,\pi(j)}| < 0$, so $\mathrm{sgn}(c^\epsilon_{\pi(j)}) = -1 = \mathrm{sgn}(c_{\pi(j)})$.
  Condition 2 implies that $\forall\ j \in [m\rangle$,

  $$|c^\epsilon_{\pi(j)}| = \mathrm{sgn}(c_{\pi(j)}) \cdot c^\epsilon_{\pi(j)}$$

- Condition 3: if $v < m-1$, then

  $$\min_{j\in[v+1\rangle} |c^\epsilon_{\pi(j)}| > \max_{j\in[m\rangle - [v+1\rangle} |c^\epsilon_{\pi(j)}|.$$

  To satisfy this condition, it is sufficient to have

  $$\epsilon_0 < \frac{|c_{\pi(0)}| - |c_{\pi(v+1)}|}{\max_{j_1\in[v+1\rangle} |g_{i^*,j_1}| + \max_{j_2\in[m\rangle - [v+1\rangle} |g_{i^*,j_2}|}$$

  if $\{j \in [m\rangle | g_{i^*,j} \neq 0\} \neq \emptyset$ (and no constraint for ϵ_0 otherwise) because then $\forall\ j_1 \in [v+1\rangle$ and $j_2 \in [m\rangle - [v+1\rangle$, $|c^\epsilon_{\pi(j_1)}| - |c^\epsilon_{\pi(j_2)}| = |c_{\pi(j_1)} + \epsilon g_{i^*,\pi(j_1)}| - |c_{\pi(j_2)} + \epsilon g_{i^*,\pi(j_2)}| \geq |c_{\pi(j_1)}| - \epsilon|g_{i^*,\pi(j_1)}| - |c_{\pi(j_2)}| - \epsilon|g_{i^*,\pi(j_2)}| = (|c_{\pi(j_1)}| - |c_{\pi(j_2)}|) - \epsilon(|g_{i^*,\pi(j_1)}| + |g_{i^*,\pi(j_2)}|) \geq (|c_{\pi(0)}| - |c_{\pi(v+1)}|) - \epsilon(|g_{i^*,\pi(j_1)}| + |g_{i^*,\pi(j_2)}|) > 0$.

Let us consider $\epsilon \in [0, \epsilon_0]$ in the following. Let a^* be the integer in $\{\pi(0), \pi(1), \cdots, \pi(v)\}$ such that

$$\mathrm{sgn}(c_{a^*}) g_{i^*,a^*} = \max_{j\in[v+1\rangle} \mathrm{sgn}(c_{\pi(j)}) g_{i^*,\pi(j)}.$$

Let

$$X^* = \{\pi(0), \pi(1), \cdots, \pi(m-1)\} - \{a^*\}.$$

Let $x_1, x_2, \cdots, x_{m-1}$ denote the $m-1$ integers in X^* such that $x_1 < x_2 < \cdots < x_{m-1}$. Let

$$s_0 = \mathrm{sgn}(c_{a^*}), \text{ and } s_j = \mathrm{sgn}(c_{x_j}) \text{ for } j = 1, 2, \cdots, m-1$$

and let

$$\psi^* = (s_0, s_1, \cdots, s_{m-1}).$$

We shall prove the following claim:

$$\mathbf{c}^\epsilon \in \mathcal{C}_{a^*,b^*,X^*,\psi^*} \text{ for all } \epsilon \in [0, \epsilon_0].$$

To prove the above claim, first, $\forall\ j \in [v+1\rangle$, we have $|c^\epsilon_{a^*}| = \mathrm{sgn}(c^\epsilon_{a^*}) \cdot c^\epsilon_{a^*} = \mathrm{sgn}(c_{a^*}) \cdot c^\epsilon_{a^*}$ (by Condition 2) $= \mathrm{sgn}(c_{a^*}) \cdot (c_{a^*} + \epsilon g_{i^*,a^*}) = \mathrm{sgn}(c_{a^*}) c_{a^*} + \epsilon \cdot \mathrm{sgn}(c_{a^*}) g_{i^*,a^*} \geq |c_{a^*}| + \epsilon \cdot \mathrm{sgn}(c_{\pi(j)}) g_{i^*,\pi(j)} = |c_{\pi(j)}| + \epsilon \cdot \mathrm{sgn}(c_{\pi(j)}) g_{i^*,\pi(j)} = \mathrm{sgn}(c_{\pi(j)}) c_{\pi(j)} + \epsilon \cdot \mathrm{sgn}(c_{\pi(j)}) g_{i^*,\pi(j)} = \mathrm{sgn}(c_{\pi(j)})(c_{\pi(j)} + \epsilon \cdot g_{i^*,\pi(j)}) = \mathrm{sgn}(c_{\pi(j)}) \cdot c^\epsilon_{\pi(j)} = \mathrm{sgn}(c^\epsilon_{\pi(j)}) \cdot c^\epsilon_{\pi(j)}$ (by Condition 2) $= |c^\epsilon_{\pi(j)}|$. Combining the above property with Condition 3, we see that $\forall\ j \in [m\rangle$, $|c^\epsilon_{a^*}| \geq |c^\epsilon_{\pi(j)}|$. Further combining it with Condition 1, we see that $\forall\ j_1 \in$

$\{1, 2, \cdots, m-1\}$ and $j_2 \in \{m, m+1, \cdots, n-1\} - \{\pi^{-1}(b^*)\}$, we get the property

$$|c^\epsilon_{a^*}| \geq |c^\epsilon_{\pi(j_1)}| > |c^\epsilon_{b^*}| \geq |c^\epsilon_{\pi(j_2)}|.$$

Second, by Condition 2, we get the property $\text{sgn}(c^\epsilon_{a^*}) = \text{sgn}(c_{a^*}) = s_0$ and for $j = 1, 2, \cdots, m-1$, $\text{sgn}(c^\epsilon_{x_j}) = \text{sgn}(c_{x_j}) = s_j$. By combining the above two properties, we prove the above claim.

Now consider $\epsilon \in (0, \epsilon_0]$ in the following, and let

$$\mathbf{d}^\epsilon = (d^\epsilon_0, d^\epsilon_1, \cdots, d^\epsilon_{n-1}) \triangleq \frac{1}{\epsilon g_{i^*,b^*}} \cdot \mathbf{c}^\epsilon.$$

Since $\mathbf{d}^\epsilon$ is just a scaled version of $\mathbf{c}^\epsilon$ and $d^\epsilon_{b^*} = c^\epsilon_{b^*}/(\epsilon g_{i^*,b^*}) = (c_{b^*} + \epsilon g_{i^*,b^*})/(\epsilon g_{i^*,b^*}) = (0 + \epsilon g_{i^*,b^*})/(\epsilon g_{i^*,b^*}) = 1$, with the same analysis as before, we get $\mathbf{d}^\epsilon \in \mathcal{Z}_{a^*,b^*,X^*,\psi^*}$ if $g_{i^*,b^*} > 0$, and $\mathbf{d}^\epsilon \in \mathcal{Z}_{a^*,b^*,X^*,-\psi^*}$ if $g_{i^*,b^*} < 0$.

Let $\hat{\mathbf{u}}^\epsilon = \frac{1}{\epsilon g_{i^*,b^*}} \cdot \mathbf{u}^\epsilon$ be the vector such that $\mathbf{d}^\epsilon = \hat{\mathbf{u}}^\epsilon G$. Then $\hat{\mathbf{u}}^\epsilon$ is a feasible solution to the linear program LP_{a^*,b^*,X^*,ψ^*} if $g_{i^*,b^*} > 0$, and is a feasible solution to the linear program $LP_{a^*,b^*,X^*,-\psi^*}$ if $g_{i^*,b^*} < 0$.

Consider the limit $\lim_{\epsilon \to 0} h_m(\mathbf{d}^\epsilon)$. It equals $\lim_{\epsilon \to 0} |d^\epsilon_{a^*}| = \lim_{\epsilon \to 0} |c^\epsilon_{a^*}|/|\epsilon g_{i^*,b^*}| = \lim_{\epsilon \to 0} |c_{a^*} + \epsilon g_{i^*,a^*}|/|\epsilon g_{i^*,b^*}| \to \infty$. So when ϵ approaches 0, the objective value of LP_{a^*,b^*,X^*,ψ^*} or $LP_{a^*,b^*,X^*,-\psi^*}$ that corresponds to the solution $\hat{\mathbf{u}}^\epsilon$ approaches ∞. So either $z_{a^*,b^*,X^*,\psi^*} = \infty$ or $z_{a^*,b^*,X^*,-\psi^*} = \infty$. Since here $m = d(\mathcal{C})$, we have $h_m(\mathcal{C}) = \infty = \max_{(a,b,X,\psi) \in \Gamma} z_{a,b,X,\psi}$. □

The above theorem naturally leads to an algorithm that computes all the m-height of a code $\mathcal{C}$, for $m = 0, 1, \cdots, n-1$, and also discovers the minimum distance $d(\mathcal{C})$:

1) $h_0(\mathcal{C}) = 1$.
2) For $m = 1, 2, 3 \cdots$, compute $h_m(\mathcal{C}) = \max_{a,b,X,\psi \in \Gamma} z_{a,b,X,\psi}$ by solving the linear programs $LP_{a,b,X,\psi}$ for all $(a, b, X, \psi) \in \Gamma$. Stop as soon as we meet the first value m^* such that $h_{m^*}(\mathcal{C}) = \infty$. We get $d(\mathcal{C}) = m^*$.
3) For $m = m^* + 1, m^* + 2, \cdots, n-1$, $h_m(\mathcal{C}) = \infty$.

The above algorithm solves $n(n-1)\binom{n-2}{m-1}2^m$ linear programs of k variables to compute an $h_m(\mathcal{C})$. That is much more efficient than the baseline algorithm, which needs to solve $n! \cdot 2^n$ linear-fractional programs of k variables. The number of (linear or linear-fractional) programs to solve is reduced by a factor of

$$\frac{n! \cdot 2^n}{n(n-1)\binom{n-2}{m-1}2^m} = (m-1)! \cdot (n-m-1)! \cdot 2^{n-m}.$$

C. Finding Exact m-Heights of Analog ECCs

The efficient m-height algorithm presented above can be used to find the exact m-heights of Analog ECCs. For some of the known codes reviewed in Section II, only upper bounds to their m-heights are known, where m is 1 or 2. For example, for the third code surveyed in Section II, which is a one-error-detecting code, it has an upper bound to its 1-height: $h_1(\mathcal{C}) \leq \lceil n/r \rceil - 1$, where $r = n - k$ is the redundancy of the $[n, k]$ code. By using the m-height algorithm, we find that for all $n \leq 50$ and $k < n$, codes built following the code construction have $h_1(\mathcal{C}) = \lceil n/r \rceil - 1$, namely, the upper bound is tight. Furthermore, we find that $h_m(\mathcal{C}) = \infty$ when $m > 1$ for those codes.

TABLE I
m-HEIGHTS OF $[n, k = n-4]$ ONE-ERROR-CORRECTING CODES

$[n,k]$	Known Bound $h_2(\mathcal{C}) \leq \lceil 2n/r \rceil - 1$	$h_0(\mathcal{C}), h_1(\mathcal{C}), \cdots, h_{n-1}(\mathcal{C})$ of built $[n,k]$ code
$[5,1]$	$h_2(\mathcal{C}) \leq 2$	1, 2, 2, 2, 2
$[6,2]$	$h_2(\mathcal{C}) \leq 2$	1, 2, 2, 3, ∞, ∞
$[7,3]$	$h_2(\mathcal{C}) \leq 3$	1, 2, 2, 3, ∞, ∞, ∞
$[8,4]$	$h_2(\mathcal{C}) \leq 3$	1, 2, 3, 3, ∞, $\cdots$, ∞
$[9,5]$	$h_2(\mathcal{C}) \leq 4$	1, 4, 4, ∞, $\cdots$, ∞
$[10,6]$	$h_2(\mathcal{C}) \leq 4$	1, 4, 4, ∞, $\cdots$, ∞
$[11,7]$	$h_2(\mathcal{C}) \leq 5$	1, 4, 4, ∞, $\cdots$, ∞
$[12,8]$	$h_2(\mathcal{C}) \leq 5$	1, 4, 5, ∞, $\cdots$, ∞

TABLE II
m-HEIGHTS OF $[n, k = n-2]$ ONE-ERROR-CORRECTING CODES

$[n,k]$	Known $h_2(\mathcal{C}) = \frac{1}{2\sin^2(\pi/(2n))} - 1$	$h_0(\mathcal{C}), h_1(\mathcal{C}), \cdots, h_{n-1}(\mathcal{C})$ of built $[n,k]$ code
$[3,1]$	$h_2(\mathcal{C}) = 1$	1, 1, 1
$[4,2]$	$h_2(\mathcal{C}) = 2.41$	1, 1.41, 2.41, ∞
$[5,3]$	$h_2(\mathcal{C}) = 4.24$	1, 2.24, 4.24, ∞, ∞
$[6,4]$	$h_2(\mathcal{C}) = 6.46$	1, 2.73, 6.46, ∞, ∞, ∞
$[7,5]$	$h_2(\mathcal{C}) = 9.10$	1, 3.49, 9.10, ∞, $\cdots$, ∞
$[8,6]$	$h_2(\mathcal{C}) = 12.14$	1, 4.03, 12.14, ∞, $\cdots$, ∞
$[9,7]$	$h_2(\mathcal{C}) = 15.58$	1, 4.76, 15.58, ∞, $\cdots$, ∞
$[10,8]$	$h_2(\mathcal{C}) = 19.43$	1, 5.31, 19.43, ∞, $\cdots$, ∞
$[11,9]$	$h_2(\mathcal{C}) = 23.69$	1, 6.03, 23.69, ∞, $\cdots$, ∞
$[12,10]$	$h_2(\mathcal{C}) = 28.35$	1, 6.60, 28.35, ∞, $\cdots$, ∞

For the fourth code surveyed in Section II, which is a one-error-correcting code, it has an upper bound to its 2-height: $h_2(\mathcal{C}) \leq \lceil 2n/r \rceil - 1$. By using the m-height algorithm, we can find its specific m-height values. Some sample results are shown in Table I, based on codes built following the code construction. (Note that the code construction in [1] is not explicit. So the codes built here may not be the only ones.) It can be seen that the upper bound for 2-height is often tight, but sometimes the actual 2-height can be lower (as for the $[7, 3]$ code and the $[11, 7]$ code). The m-height algorithm also discovers the m-heights of the codes for $m \neq 2$ as well.

For the sixth code surveyed in Section II, which is a one-error-correcting code, it has a known formula for its 2-height: $h_2(\mathcal{C}) = \frac{1}{2\sin^2(\pi/(2n))} - 1$. Using the m-height algorithm, we can find its m-heights for $m \neq 2$ as well. Some sample results are shown in Table II.

IV. ANALOG PERMUTATION CODE

The design of Analog ECCs is still a largely open problem. As can be seen from the survey in Section II, existing codes are mainly limited to detecting or correcting one UME (except for repetition codes). So it is important to explore various approaches for designing new codes. One fundamental method of code design is through computer search, where good codes are identified from many randomly sampled codes (possibly with guidance on their code constructions). To analyze the error-correction capabilities of the codes, their m-heights

need to be computed. It is therefore important to make the computing of m-heights as efficient as possible, so that more codes can be sampled and analyzed. One way to reduce the complexity of the m-height algorithm is to construct codes with special structures. In this section, we explore one such code structure, where the columns of the generator matrix are distinct permutations of the same set of real numbers. Let us call such codes *Analog Permutation Codes*. The complexity of the m-height algorithm for $[n, k]$ Analog Permutation Codes can be reduced by a factor of n.

A. Construction and m-Height of Analog Permutation Code

Consider linear $[n, k]$ Analog ECCs over $\mathbb{R}$, where $n = k!$. Let $\pi : [k\rangle \rightarrow [k\rangle$ denote a permutation on $[k\rangle$. Let Π denote the set of all $k!$ such permutations. Let us label those $k!$ permutations by $\pi_0, \pi_1, \cdots, \pi_{k!-1}$, namely, $\Pi = \{\pi_i | i \in [k!\rangle\}$. For convenience, let π_0 be the identity permutation, namely, $\pi_0(i) = i$ for all $i \in [k\rangle$.

Similarly, let $\lambda : [n\rangle \rightarrow [n\rangle$ denote a permutation on $[n\rangle$. (Note that $n = k!$.) Let Λ denote the set of all $n!$ such permutations. Let us label those $n!$ permutations by $\lambda_0, \lambda_1, \cdots, \lambda_{n!-1}$, namely, $\Lambda = \{\lambda_i | i \in [n!\rangle\}$. For convenience, let λ_0 be the identity permutation, namely, $\lambda_0(i) = i$ for all $i \in [n\rangle$.

Let $\mathbf{g} = (g_0, g_1, \cdots, g_{k-1})^T \in \mathbb{R}^k$ be a column vector of length k. Given a permutation $\pi \in \Pi$, let

$$\mathbf{g}_\pi \triangleq (g_{\pi(0)}, g_{\pi(1)}, \cdots, g_{\pi(k-1)})^T \in \mathbb{R}^k$$

be a column vector that permutes the entries in $\mathbf{g}$ by the permutation π. Define a $k \times n$ matrix G as

$$G = (\mathbf{g}_{\pi_0}, \mathbf{g}_{\pi_1}, \cdots, \mathbf{g}_{\pi_{n-1}}).$$

An Analog ECC with G as its generator matrix is called an *Analog Permutation Code*.

$\forall\, \pi \in \Pi$, let us use $(\pi(0), \pi(1), \cdots, \pi(k-1))$ to denote the value of the permutation π. (For example, $\pi = (2, 0, 1)$ represents a permutation π where $\pi(0) = 2$, $\pi(1) = 0$ and $\pi(2) = 1$. Note that this notation is different from the *cyclic form* of permutations.) Similarly, we use $(\lambda(0), \lambda(1), \cdots, \lambda(n-1))$ to denote the value of a permutation $\lambda \in \Lambda$.

Example 1: Let $k = 3$ and $n = k! = 6$. Let $\pi_0 = (0, 1, 2)$, $\pi_1 = (0, 2, 1)$, $\pi_2 = (1, 0, 2)$, $\pi_3 = (1, 2, 0)$, $\pi_4 = (2, 0, 1)$, $\pi_5 = (2, 1, 0)$. Then the $k \times n$ generator matrix G is

$$G = \begin{pmatrix} g_0 & g_0 & g_1 & g_1 & g_2 & g_2 \\ g_1 & g_2 & g_0 & g_2 & g_0 & g_1 \\ g_2 & g_1 & g_2 & g_0 & g_1 & g_0 \end{pmatrix}$$

□

In this section, we shall prove the following result on the m-height algorithm for Analog Permutation Codes.

Theorem 3: Let $\mathcal{C}$ be a linear $[n, k]$ Analog Permutation Code over $\mathbb{R}$. Let G be its generator matrix where no column is $\mathbf{0}$. Let $d(\mathcal{C})$ be the minimum distance of $\mathcal{C}$, and let $m \in \{1, 2, \cdots, \min\{d(\mathcal{C}), n-1\}\}$. Let $z_{a,b,X,\psi}$ be as defined in Theorem 2. Then

$$h_m(\mathcal{C}) = \max_{(0,b,X,\psi)\in\Gamma} z_{0,b,X,\psi}$$

Compared to Theorem 2, the theorem above improves the time complexity of the m-height algorithm by a factor of n. Instead of solving linear programs for all the tuples $(a, b, X, \psi) \in \Gamma$, here we fix a to 0, and consider only those tuples $(0, b, X, \psi)$. To prove the theorem, let us start by analyzing properties of Analog Permutation Codes.

B. Properties of Analog Permutation Code

Given an *information vector* $\mathbf{u} = (u_0, u_1, \cdots, u_{k-1}) \in \mathbb{R}^k$ and the generator matrix G, let $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) = \mathbf{u}G = (\mathbf{u}\mathbf{g}_{\pi_0}, \mathbf{u}\mathbf{g}_{\pi_1}, \cdots, \mathbf{u}\mathbf{g}_{\pi_{n-1}})$ be the corresponding codeword. Given any $i \in [k!\rangle$, define $\mathbf{u}_{\pi_i}$ as

$$\mathbf{u}_{\pi_i} \triangleq (u_{\pi_i(0)},\ u_{\pi_i(1)},\ \cdots,\ u_{\pi_i(k-1)}),$$

which permutes the k numbers of the vector $\mathbf{u}$ by the permutation $\pi_i \in \Pi$. Given any $j \in [n!\rangle$, define G_{λ_j} as

$$G_{\lambda_j} \triangleq (\mathbf{g}_{\pi_{\lambda_j(0)}},\ \mathbf{g}_{\pi_{\lambda_j(1)}},\ \cdots,\ \mathbf{g}_{\pi_{\lambda_j(n-1)}}),$$

which permutes the n columns of the generator matrix G by the permutation $\lambda_j \in \Lambda$. Given any $j \in [n!\rangle$, define $\mathbf{c}_{\lambda_j}$ as

$$\mathbf{c}_{\lambda_j} \triangleq (c_{\lambda_j(0)},\ c_{\lambda_j(1)},\ \cdots,\ c_{\lambda_j(n-1)}),$$

which permutes the n numbers of the codeword $\mathbf{c}$ by the permutation $\lambda_j \in \Lambda$. We have $\mathbf{c}_{\lambda_j} = \mathbf{u}G_{\lambda_j}$ for $j \in [n!\rangle$.

$\forall\ i \in [k!\rangle$, $j \in [n!\rangle$ and $p \in [n\rangle$, let $f_{i,j}(p)$ be the integer in $[n\rangle$ such that

$$\mathbf{u}_{\pi_i}\mathbf{g}_{\pi_{\lambda_j(p)}} \equiv \mathbf{u}\mathbf{g}_{\pi_{f_{i,j}(p)}}. \tag{1}$$

Since $\mathbf{u}_{\pi_i}\mathbf{g}_{\pi_{\lambda_j(p)}} = \sum_{z\in[k\rangle} u_{\pi_i(z)} g_{\pi_{\lambda_j(p)}(z)} = \sum_{z\in[k\rangle} u_z g_{\pi_{\lambda_j(p)}(\pi_i^{-1}(z))}$, we see that $f_{i,j}(p)$ is the integer in $[n\rangle$ such that

$$\pi_{f_{i,j}(p)} = \pi_{\lambda_j(p)} \circ \pi_i^{-1} \tag{2}$$

where "$\circ$" is the "composite" operation between two functions (namely, for any two functions h_1 and h_2, their composite $h_1 \circ h_2$ is a function such that $(h_1 \circ h_2)(x) = h_1(h_2(x))$ for all x), and π_i^{-1} is the inverse function of π_i.

Lemma 2: $\forall\ i \in [k!\rangle$ and $j \in [n!\rangle$, the function $f_{i,j}$, which maps p to $f_{i,j}(p)$ for all $p \in [n\rangle$, is a permutation on $[n\rangle$.

Proof: Since $\pi_{\lambda_j(p)} \circ \pi_i^{-1}$ is a permutation on $[k\rangle$, $\pi_{f_{i,j}(p)}$ is a permutation on $[k\rangle$. So $f_{i,j}(p) \in [k!\rangle = [n\rangle$. $\forall\ p \neq q \in [n\rangle$, $\lambda_j(p) \neq \lambda_j(q)$ as λ_j is a permutation, so $\pi_{\lambda_j(p)} \neq \pi_{\lambda_j(q)}$ and therefore $\pi_{\lambda_j(p)} \circ \pi_i^{-1} \neq \pi_{\lambda_j(q)} \circ \pi_i^{-1}$. So whenever $p \neq q$, $\pi_{f_{i,j}(p)} \neq \pi_{f_{i,j}(q)}$ and therefore $f_{i,j}(p) \neq f_{i,j}(q)$. So $f_{i,j}$ is a permutation on $[n\rangle$. □

$\forall\ j \in [n!\rangle$, let S_j be a set of $n = k!$ permutations on $[n\rangle$:

$$S_j \triangleq \{f_{i,j} \mid i \in [k!\rangle\}.$$

Example 2: Let us continue with Example 1. Let $j = 0$. Since λ_0 is the identity permutation, $\pi_{\lambda_j(p)} = \pi_p$. Now let us find $f_{i,0}$ for $i = 0, 1, \cdots, k! - 1 = 5$. Let us show how to compute $f_{i,0}$ for $i = 3$. The other $f_{i,j}$'s can be computed in a similar way.

When $i = 3$, we have $\pi_3 = (1, 2, 0)$, therefore $\pi_3^{-1} = (2, 0, 1)$. When $p = 0$, since $\mathbf{u}_{\pi_i}\mathbf{g}_{\pi_{\lambda_j(p)}} =$

$\mathbf{u}_{\pi_3}\mathbf{g}_{\pi_0} = (u_{\pi_3(0)}, u_{\pi_3(1)}, u_{\pi_3(2)}) \cdot (g_{\pi_0(0)}, g_{\pi_0(1)}, g_{\pi_0(2)})^T = (u_1, u_2, u_0) \cdot (g_0, g_1, g_2)^T = (u_0, u_1, u_2) \cdot (g_2, g_0, g_1)^T = \mathbf{u}\mathbf{g}_{\pi_4}$, we get $f_{i,j}(p) = f_{3,0}(0) = 4$. (Another way to compute $f_{3,0}(0)$ is that since $\pi_{f_{i,j}(p)} = \pi_{\lambda_j(p)} \circ \pi_i^{-1}$, we get $\pi_{f_{3,0}(0)} = \pi_{\lambda_0(0)} \circ \pi_3^{-1} = \pi_0 \circ \pi_3^{-1} = (0,1,2) \circ (2,0,1) = (2,0,1) = \pi_4$, therefore $f_{3,0}(0) = 4$.) In the same way, we get $f_{3,0}(1) = 2$, $f_{3,0}(2) = 5$, $f_{3,0}(3) = 0$, $f_{3,0}(4) = 3$, $f_{3,0}(5) = 1$. So the permutation $f_{3,0} = (4, 2, 5, 0, 3, 1)$.

In the same way, we get $f_{0,0} = (0, 1, 2, 3, 4, 5)$, $f_{1,0} = (1, 0, 3, 2, 5, 4)$, $f_{2,0} = (2, 4, 0, 5, 1, 3)$, $f_{4,0} = (3, 5, 1, 4, 0, 2)$, $f_{5,0} = (5, 3, 4, 1, 2, 0)$. And $S_0 = \{f_{0,0}, f_{1,0}, \cdots, f_{5,0}\}$. □

Lemma 3: The n permutations in the set S_0 are all different. That is, $\forall\ i \neq j \in [n\rangle$, $f_{i,0} \neq f_{j,0}$.

Proof: Consider $f_{i,0}(0)$ and $f_{j,0}(0)$. Since $\pi_{f_{i,0}(0)} = \pi_{\lambda_0(0)} \cdot \pi_i^{-1} = \pi_0 \cdot \pi_i^{-1} = \pi_i^{-1}$ and similarly $\pi_{f_{j,0}(0)} = \pi_j^{-1}$, we get $\pi_{f_{i,0}(0)} \neq \pi_{f_{j,0}(0)}$ and therefore $f_{i,0}(0) \neq f_{j,0}(0)$. Since two functions are equal only if they have the same value for every input, we get $f_{i,0} \neq f_{j,0}$. □

It is worthwhile to note that $f_{i,j}$, as a permutation on $[n\rangle$:

$$f_{i,j} = \begin{pmatrix} 0 & 1 & \cdots & n-1 \\ f_{i,j}(0) & f_{i,j}(1) & \cdots & f_{i,j}(n-1) \end{pmatrix}$$

can be equivalently seen as a permutation on $\{\pi_0, \pi_1, \cdots, \pi_{n-1}\}$:

$$f_{i,j} = \begin{pmatrix} \pi_0 & \pi_1 & \cdots & \pi_{n-1} \\ \pi_{f_{i,j}(0)} & \pi_{f_{i,j}(1)} & \cdots & \pi_{f_{i,j}(n-1)} \end{pmatrix}.$$

We shall call the above the *equivalent view* of $f_{i,j}$. Since $\pi_{f_{i,j}(p)} = \pi_{\lambda_j(p)} \circ \pi_i^{-1}$ by Equation 2, the *equivalent view* of $f_{i,j}$ can also be expressed as:

$$\begin{pmatrix} \pi_0 & \pi_1 & \cdots & \pi_{n-1} \\ \pi_{\lambda_j(0)} \circ \pi_i^{-1} & \pi_{\lambda_j(1)} \circ \pi_i^{-1} & \cdots & \pi_{\lambda_j(n-1)} \circ \pi_i^{-1} \end{pmatrix}.$$

Lemma 4: $\langle S_0, \circ\rangle$ is a permutation group.

Proof: The "composite" operation $\circ$ is known to be associative. Let us now show that S_0 is closed under the operation $\circ$. Consider any two permutations $f_{i,0} \in S_0$ and $f_{j,0} \in S_0$ and their *equivalent views*. $\forall\ p \in [n\rangle$, since $\pi_{f_{j,0}(p)} = \pi_{\lambda_0(p)} \circ \pi_j^{-1} = \pi_p \circ \pi_j^{-1}$, the permutation $f_{j,0}$ maps π_p to $\pi_p \circ \pi_j^{-1}$. Similarly, since $\pi_{f_{i,0}(q)} = \pi_q \circ \pi_i^{-1}$ for all $q \in [n\rangle$, the permutation $f_{i,0}$ maps π_q to $\pi_q \circ \pi_i^{-1}$ for all $\pi_q \in \Pi$. So the permutation $f_{i,0} \circ f_{j,0}$ maps π_p to $(\pi_p \circ \pi_j^{-1}) \circ \pi_i^{-1} = \pi_p \circ (\pi_j^{-1} \circ \pi_i^{-1}) = \pi_p \circ (\pi_i \circ \pi_j)^{-1}$. Define $\pi_{\varpi(i,j)} \triangleq \pi_i \circ \pi_j \in \Pi$. Then $f_{i,0} \circ f_{j,0}$ maps π_p to $\pi_p \circ \pi_{\varpi(i,j)}^{-1}$ for all $p \in [n\rangle$. So $f_{i,0} \circ f_{j,0} = f_{\varpi(i,j),0} \in S_0$. Therefore S_0 is closed under the operation $\circ$.

Let us show that $f_{0,0} \in S_0$ is an identity element, namely, for any $f_{i,0} \in S_0$, $f_{i,0} \circ f_{0,0} = f_{0,0} \circ f_{i,0} = f_{i,0}$. By the analysis above, $f_{0,0}$ maps π_p to $\pi_p \circ \pi_0^{-1} = \pi_p$ for all $p \in [n\rangle$. So $f_{0,0}$ is the identity permutation $(0, 1, \cdots .n-1)$. So $f_{0,0}$ is clearly an identity element in S_0.

Let us now show that every permutation $f_{i,0} \in S_0$ has an inverse. For all $i \in [n\rangle$, let $\varphi(i) \in [n\rangle$ be the integer such that $\pi_{\varphi(i)} = \pi_i^{-1}$. By the above analysis, $\forall\ p \in [n\rangle$, $f_{i,0} \circ f_{\varphi(i),0}$ maps π_p to $\pi_p \circ (\pi_{\varphi(i)}^{-1} \circ \pi_i^{-1}) = \pi_p \circ (\pi_i \circ \pi_i^{-1}) = \pi_p$, and $f_{\varphi(i),0} \circ f_{i,0}$ maps π_p to $\pi_p \circ (\pi_i^{-1} \circ \pi_{\varphi(i)}^{-1}) = \pi_p \circ (\pi_i^{-1} \circ \pi_i) = \pi_p$. So $f_{i,0} \circ f_{\varphi(i),0} = f_{\varphi(i),0} \circ f_{i,0} = (0, 1, \cdots, n-1) = f_{0,0}$. So $f_{i,0} \in S_0$ has an inverse $f_{\varphi(i),0} \in S_0$. Therefore $\langle S_0, \circ\rangle$ is a permutation group. □

The group S_0 is a subgroup of the symmetric group of $[n\rangle$, $\mathrm{Sym}([n\rangle)$, which contains all the $n!$ permutations on $[n\rangle$. There are $(n-1)!$ *distinct* right cosets of S_0 in $\mathrm{Sym}([n\rangle)$.

Lemma 5: $\forall\ i \in [k!\rangle$ and $j \in [n!\rangle$, $f_{i,j} = f_{i,0} \circ f_{0,j}$. Also, $\forall\ j \in [n!\rangle$, S_j is a right coset of S_0, and $S_j = S_0 \circ f_{0,j}$.

Proof: $\forall\ i \in [k!\rangle$, $f_{i,0}$ is in S_0, and by its *equivalent view*, the permutation $f_{i,0}$ maps π_p to $\pi_p \circ \pi_i^{-1}$ for all $p \in [n\rangle$. $f_{0,j}$ is in $\mathrm{Sym}([n\rangle)$, and since $\pi_{f_{0,j}(q)} = \pi_{\lambda_j(q)} \circ \pi_0^{-1} = \pi_{\lambda_j(q)}$, it maps π_q to $\pi_{\lambda_j(q)}$ for all $q \in [n\rangle$. So for all $p \in [n\rangle$, the permutation $f_{i,0} \circ f_{0,j}$ maps π_p to $\pi_{\lambda_j(p)} \circ \pi_i^{-1} = \pi_{f_{i,j}(p)}$. So $f_{i,0} \circ f_{0,j} = f_{i,j}$, and that leads to the conclusion. □

Let $\mathcal{C}$ be a linear $[n, k]$ Analog Permutation Code, whose generator matrix is G. Given a permutation $\lambda \in \Lambda$ on $[n\rangle$, we define $\mathcal{C}_\lambda$ as the following subset of codewords:

$$\mathcal{C}_\lambda \triangleq \{(c_0, c_1, \cdots, c_{n-1}) \in \mathcal{C}\ :\ |c_{\lambda(0)}| \geq |c_{\lambda(1)}| \geq \cdots \geq |c_{\lambda(n-1)}|\}.$$

As before, the m-height of $\mathcal{C}_\lambda$ is defined as

$$h_m(\mathcal{C}_\lambda) = \max_{\mathbf{c} \in \mathcal{C}_\lambda} h_m(\mathbf{c})$$

if $\mathcal{C}_\lambda \neq \emptyset$, and $h_m(\mathcal{C}_\lambda) = 0$ otherwise.

Lemma 6: $\forall\ j \in [n!\rangle$, $p \in [k!\rangle$, $q \in [k!\rangle$ and $m \in [n\rangle$,

$$h_m(\mathcal{C}_{f_{p,j}}) = h_m(\mathcal{C}_{f_{q,j}}).$$

Proof: Let us first assume that $\mathcal{C}_{f_{p,j}} \neq \emptyset$. In this case, let $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1})$ be a codeword in $\mathcal{C}_{f_{p,j}}$ such that $h_m(\mathbf{c}) = h_m(\mathcal{C}_{f_{p,j}})$. Let $\mathbf{u} = (u_0, u_1, \cdots, u_k) \in \mathbb{R}^k$ be the vector such that $\mathbf{c} = \mathbf{u}G$, where $G = (\mathbf{g}_{\pi_0}, \mathbf{g}_{\pi_1}, \cdots, \mathbf{g}_{\pi_{n-1}})$ is the generator matrix.

Define $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ as $\mathbf{v} = \mathbf{u}_{\pi_p} G_{\lambda_j}$, where $G_{\lambda_j} = (\mathbf{g}_{\pi_{\lambda_j(0)}}, \mathbf{g}_{\pi_{\lambda_j(1)}}, \cdots, \mathbf{g}_{\pi_{\lambda_j(n-1)}})$ permutes the columns of G by λ_j. (Note that $\mathbf{v}$ may not be a codeword.) Then $\forall\ i \in [n\rangle$, $v_i = \mathbf{u}_{\pi_p}\mathbf{g}_{\pi_{\lambda_j(i)}}$. By Equation 1, $v_i = \mathbf{u}\mathbf{g}_{\pi_{f_{p,j}(i)}}$. Since $c_i = \mathbf{u}\mathbf{g}_{\pi_i}$, $v_i = c_{f_{p,j}(i)}$. So $\mathbf{v} = (c_{f_{p,j}(0)}, c_{f_{p,j}(1)}, \cdots, c_{f_{p,j}(n-1)})$, which permutes the numbers in $\mathbf{c}$ by the permutation $f_{p,j}$. Since $\mathbf{c} \in \mathcal{C}_{f_{p,j}}$, $|c_{f_{p,j}(0)}| \geq |c_{f_{p,j}(1)}| \geq \cdots \geq |c_{f_{p,j}(n-1)}|$. So $|v_0| \geq |v_1| \geq \cdots \geq |v_{n-1}|$. So $h_m(\mathbf{c}) = |v_0|/|v_m|$ if $v_m \neq 0$, $h_m(\mathbf{c}) = \infty$ if $v_m = 0$ and $v_0 \neq 0$, and $h_m(\mathbf{c}) = 0$ otherwise (as in this case $\mathbf{c} = \mathbf{0}$).

Define $\mathbf{a} = (a_0, a_1, \cdots, a_{k-1}) \in \mathbb{R}^k$ as $\mathbf{a} = \mathbf{u}_{\pi_p \circ \pi_q^{-1}}$. Define a codeword $\mathbf{b} = (b_0, b_1, \cdots, b_{n-1})$ as $\mathbf{b} = \mathbf{a}G$. Since $\mathbf{a}_{\pi_q} = (a_{\pi_q(0)}, a_{\pi_q(1)}, \cdots, a_{\pi_q(n-1)})$ and $a_i = u_{\pi_p \circ \pi_q^{-1}(i)}$ for all i, we get $a_{\pi_q(i)} = u_{\pi_p \circ \pi_q^{-1} \circ \pi_q(i)} = u_{\pi_p(i)}$, and therefore $\mathbf{a}_{\pi_q} = (u_{\pi_p(0)}, u_{\pi_p(1)}, \cdots, u_{\pi_p(n-1)}) = \mathbf{u}_{\pi_p}$. So $\mathbf{a}_{\pi_q} G_{\lambda_j} = \mathbf{u}_{\pi_p} G_{\lambda_j} = \mathbf{v}$. So $\forall\ i \in [n\rangle$, $v_i = \mathbf{a}_{\pi_q}\mathbf{g}_{\pi_{\lambda_j(i)}} = \mathbf{a}\mathbf{g}_{\pi_{f_{q,j}(i)}}$. Therefore $\mathbf{v} = (\mathbf{a}\mathbf{g}_{\pi_{f_{q,j}(0)}}, \mathbf{a}\mathbf{g}_{\pi_{f_{q,j}(1)}}, \cdots, \mathbf{a}\mathbf{g}_{\pi_{f_{q,j}(n-1)}})$. Since $\mathbf{b} = \mathbf{a}G = (\mathbf{a}\mathbf{g}_{\pi_0}, \mathbf{a}\mathbf{g}_{\pi_1}, \cdots, \mathbf{a}\mathbf{g}_{\pi_{n-1}})$, $\mathbf{v}$ permutes the n numbers in the codeword $\mathbf{b}$ by the permutation $f_{q,j}$. Since $|v_0| \geq |v_1| \geq \cdots \geq |v_{n-1}|$, we get $|\mathbf{a}\mathbf{g}_{\pi_{f_{q,j}(0)}}| \geq |\mathbf{a}\mathbf{g}_{\pi_{f_{q,j}(1)}}| \geq \cdots \geq |\mathbf{a}\mathbf{g}_{\pi_{f_{q,j}(n-1)}}|$, therefore $|b_{f_{q,j}(0)}| \geq |b_{f_{q,j}(1)}| \geq \cdots \geq |b_{f_{q,j}(n-1)}|$, which means $\mathbf{b} \in \mathcal{C}_{f_{q,j}}$. Furthermore, $h_m(\mathbf{b}) = h_m(\mathbf{c})$ because both $\mathbf{b}$ and $\mathbf{c}$ are permutations of $\mathbf{v}$. So $h_m(\mathcal{C}_{f_{p,j}}) = h_m(\mathbf{c}) = h_m(\mathbf{b}) \leq \max_{\mathbf{b}' \in \mathcal{C}_{f_{q,j}}} h_m(\mathbf{b}') = h_m(\mathcal{C}_{f_{q,j}})$. In the same way, we can

prove $h_m(\mathcal{C}_{f_{q,j}}) \leq h_m(\mathcal{C}_{f_{p,j}})$. So $h_m(\mathcal{C}_{f_{p,j}}) = h_m(\mathcal{C}_{f_{q,j}})$ when $\mathcal{C}_{f_{p,j}} \neq \emptyset$.

When $\mathcal{C}_{f_{p,j}} = \emptyset$, by the above analysis, it is easy to see $\mathcal{C}_{f_{q,j}} = \emptyset$, too. In this case, $h_m(\mathcal{C}_{f_{p,j}}) = h_m(\mathcal{C}_{f_{q,j}}) = 0$. □

Lemma 7: $\forall\ j \in [n!\rangle$, $f_{\lambda_j(0),j}(0) = 0$.

Proof: By its *equivalent view*, the permutation $f_{\lambda_j(0),j}$ maps π_0 to $\pi_{f_{\lambda_j(0),j}(0)} = \pi_{\lambda_j(0)} \circ \pi^{-1}_{\lambda_j(0)} = \pi_0$. (The first equality is due to Equation 2.) So $f_{\lambda_j(0),j}$ maps 0 to 0 by its original definition, namely, $f_{\lambda_j(0),j}(0) = 0$. □

Theorem 4: Let $j_0, j_1, \cdots, j_{(n-1)!-1}$ be any $(n-1)!$ integers in $[n!\rangle$ such that $S_{j_0}, S_{j_1}, \cdots, S_{j_{(n-1)!-1}}$ are $(n-1)!$ distinct right cosets of S_0. Then $\forall\ m \in [n\rangle$,

$$h_m(\mathcal{C}) = \max_{i \in [(n-1)!\rangle} h_m(\mathcal{C}_{f_{\lambda_{j_i}(0),j_i}}).$$

Proof: Since $\mathcal{C} = \bigcup_{\lambda \in \Lambda} \mathcal{C}_\lambda$, we get $h_m(\mathcal{C}) = \max_{\lambda \in \Lambda} h_m(\mathcal{C}_\lambda)$. Since $S_{j_0}, S_{j_1}, \cdots, S_{j_{(n-1)!-1}}$ form a partition of Λ, we get $h_m(\mathcal{C}) = \max_{i \in [(n-1)!\rangle} \max_{\lambda \in S_{j_i}} h_m(\mathcal{C}_\lambda)$. Since $S_{j_i} = \{f_{p,j_i} | p \in [k!\rangle\}$, we get $h_m(\mathcal{C}) = \max_{i \in [(n-1)!\rangle} \max_{p \in [k!\rangle} h_m(\mathcal{C}_{f_{p,j_i}})$. By Lemma 6, $h_m(\mathcal{C}_{f_{p,j_i}}) = h_m(\mathcal{C}_{f_{\lambda_{j_i}(0),j_i}})$, so we get $h_m(\mathcal{C}) = \max_{i \in [(n-1)!\rangle} h_m(\mathcal{C}_{f_{\lambda_{j_i}(0),j_i}})$. □

The result below connects the properties of Analog Permutations Codes shown above with the concepts used to compute the m-heights of analog codes in Section III.

Corollary 1: Let $\mathcal{C}$ be a linear $[n = k!, k]$ Analog Permutation Code. Then

$$h_m(\mathcal{C}) = \max_{(0,b,X,\psi) \in \Gamma} h_m(\mathcal{C}_{0,b,X,\psi}).$$

Proof: Consider any codeword $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in \mathcal{C}_{f_{\lambda_j(0),j}} - \{\mathbf{0}\}$ for any $j \in [n!\rangle$. By definition, $|c_{f_{\lambda_j(0),j}(0)}| \geq |c_{f_{\lambda_j(0),j}(1)}| \geq \cdots \geq |c_{f_{\lambda_j(0),j}(n-1)}|$. Since $f_{\lambda_j(0),j}(0) = 0$ by Lemma 7, we get $|c_0| \geq |c_{f_{\lambda_j(0),j}(1)}| \geq \cdots \geq |c_{f_{\lambda_j(0),j}(n-1)}|$. So there exists a tuple $(0, b, X, \psi)$ such that $\mathbf{c} \in \mathcal{C}_{0,b,X,\psi}$.

$h_m(\mathcal{C}) = \max_{i \in [(n-1)!\rangle} h_m(\mathcal{C}_{f_{\lambda_{j_i}(0),j_i}})$ by Theorem 4. So $h_m(\mathcal{C}) = \max_{i \in [(n-1)!\rangle} \max_{\mathbf{c} \in \mathcal{C}_{f_{\lambda_{j_i}(0),j_i}}} h_m(\mathbf{c})$. By the analysis above, $\mathcal{C}_{f_{\lambda_{j_i}(0),j_i}} - \{\mathbf{0}\} \subseteq \bigcup_{(0,b,X,\psi) \in \Gamma} \mathcal{C}_{0,b,X,\psi}$ for every j_i. So $h_m(\mathcal{C} - \{\mathbf{0}\}) \leq h_m(\bigcup_{(0,b,X,\psi) \in \Gamma} \mathcal{C}_{0,b,X,\psi})$. Since $h_m(\mathbf{0}) = 0$, we get $h_m(\mathcal{C}) \leq \max_{(0,b,X,\psi) \in \Gamma} h_m(\mathcal{C}_{0,b,X,\psi})$. Since $\mathcal{C}_{0,b,X,\psi} \subseteq \mathcal{C}$, we get $h_m(\mathcal{C}) = \max_{(0,b,X,\psi) \in \Gamma} h_m(\mathcal{C}_{0,b,X,\psi})$. □

C. m-Height Algorithm for Analog Permutation Code

We now prove Theorem 3, which reduces the complexity of computing the m-height of an $[n, k]$ Analog Permutation Code by a factor of n compared to Theorem 2.

Proof of Theorem 3: There are two cases to consider: $m < d(\mathcal{C})$ and $m = d(\mathcal{C})$. In the case of $m < d(\mathcal{C})$, the proof is essentially the same as that of Theorem 3. (We just need to replace (a, b, X, ψ) by $(0, b, X, \psi)$ in the proof. Recall that by Corollary 1, $h_m(\mathcal{C}) = \max_{(0,b,X,\psi) \in \Gamma} h_m(\mathcal{C}_{0,b,X,\psi})$ instead of $\max_{(a,b,X,\psi) \in \Gamma} h_m(\mathcal{C}_{a,b,X,\psi})$.) So we skip the details here.

Let us now consider the case $m = d(\mathcal{C})$. In this case, the proof of Theorem 2 shows that there exists a vector $\mathbf{u} = (u_0, u_1, \cdots, u_{k-1}) \in \mathbb{R}^k$, an integer $i^* \in [k\rangle$, and a positive constant ϵ_0 such that:

Algorithm 1 A Genetic Programming Algorithm for Constructing Analog ECCs

Input : Integers k, n, m, N, N_c, N_p, N_t in $\mathbb{Z}^+$ with $n > k$, $1 \leq m \leq n - k$, $N < N_c$; a small number $\delta \in \mathbb{R}^+$

Output: A set of N generator matrices of Analog ECCs

Initialization: let S be a set of N_c randomly generated $k \times n$ generator matrices, where for each generator matrix, its independent numbers follow the $\mathcal{N}(0, 1)$ Gaussian distribution. (If the generator matrix is for a permutation code or a punctured permutation code, only the k numbers in a column are independent; otherwise, all the nk numbers in the matrix are independent.);
Compute the m-height of each code whose generator matrix is in S;
$S \leftarrow$ the N generator matrices in S whose m-heights are the smallest;
for $i \leftarrow 1$ **to** N_p **do**
  Let S_{new} be an empty set;
  for $j \leftarrow 1$ **to** N_t **do**
    Mutation step: Uniformly randomly select a generator matrix G from S;
    Let P be a randomly generated $k \times n$ matrix whose nk numbers follow the $\mathcal{N}(0, 1)$ Gaussian distribution;
    $\sigma \leftarrow \delta$;
    $G_{new} \leftarrow G + \sigma P$;
    while *m-height of G_{new} $<$ m-height of G* **do**
      $S_{new} \leftarrow S_{new} \cup \{G_{new}\}$;
      $G \leftarrow G_{new}$;
      $\sigma \leftarrow 2\sigma$;
      $G_{new} \leftarrow G + \sigma P$;
    Crossover step: Uniformly randomly select two generator matrices G_1 and G_2 from S;
    Let G_3 be a $k \times n$ matrix half of whose columns are randomly chosen from G_1 and the other half from G_2;
    if *m-height of G_3 is less than the m-heights of both G_1 and G_2* **then**
      $S_{new} \leftarrow S_{new} \cup \{G_3\}$;
  $S \leftarrow S \cup S_{new}$;
  $S \leftarrow$ the N generator matrices in S whose m-heights are the smallest;
return S

- Property: Let $\mathbf{u}^\epsilon = (u_0^\epsilon, u_1^\epsilon, \cdots, u_{k-1}^\epsilon) \triangleq (u_0, \cdots, u_{i^*-1}, u_{i^*} + \epsilon, u_{i^*+1}, \cdots, u_{k-1})$, and let $\mathbf{c}^\epsilon = (c_0^\epsilon, c_1^\epsilon, \cdots, c_{n-1}^\epsilon) \triangleq \mathbf{u}^\epsilon G$. When $\epsilon \in [0, \epsilon_0]$, there exists a tuple (a^*, b^*, X^*, ψ^*) with $|X^*| = m - 1$ such that

$$\mathbf{c}^\epsilon \in \mathcal{C}_{a^*,b^*,X^*,\psi^*} \text{ for all } \epsilon \in (0, \epsilon_0].$$

Let $\bar{a} \in [n\rangle$ be the integer such that $\pi_{\bar{a}} = \pi_{a^*}^{-1}$. Now consider the vector $\mathbf{u}^{\epsilon}_{\pi_{\bar{a}}} = (u^{\epsilon}_{\pi_{\bar{a}}(0)}, u^{\epsilon}_{\pi_{\bar{a}}(1)}, \cdots, u^{\epsilon}_{\pi_{\bar{a}}(k-1)})$ and its corresponding codeword $\bar{\mathbf{c}}^{\epsilon} = (\bar{c}^{\epsilon}_0, \bar{c}^{\epsilon}_1, \cdots, \bar{c}^{\epsilon}_{n-1}) \triangleq \mathbf{u}^{\epsilon}_{\pi_{\bar{a}}} G$. Note that since λ_0 is the identity permutation, $G = G_{\lambda_0}$. So $\bar{\mathbf{c}}^{\epsilon} = \mathbf{u}^{\epsilon}_{\pi_{\bar{a}}} G_{\lambda_0} = \mathbf{u}^{\epsilon}_{\pi_{\bar{a}}} (\mathbf{g}_{\pi_{\lambda_0(0)}}, \mathbf{g}_{\pi_{\lambda_0(1)}}, \cdots, \mathbf{g}_{\pi_{\lambda_0(n-1)}})$. So $\forall\ p \in [n\rangle$, we have

$$\bar{c}^{\epsilon}_p = \mathbf{u}^{\epsilon}_{\pi_{\bar{a}}} \mathbf{g}_{\pi_{\lambda_0(p)}} = \mathbf{u}^{\epsilon} \mathbf{g}_{\pi_{f_{\bar{a},0}(p)}} = c^{\epsilon}_{f_{\bar{a},0}(p)}.$$

(The second equality is due to Equation 1.) Since $f_{\bar{a},0}$ is a permutation on $[n\rangle$, the n numbers in the codeword $\bar{\mathbf{c}}^{\epsilon}$ form a permutation of the n numbers in the codeword $\mathbf{c}^{\epsilon}$.

Since $\pi_{f_{\bar{a},0}(p)} = \pi_{\lambda_0(p)} \circ \pi_{\bar{a}}^{-1} = \pi_p \circ \pi_{\bar{a}}^{-1}$ due to Equation 2, by letting $p = 0$, we get $\pi_{f_{\bar{a},0}(0)} = \pi_0 \circ \pi_{\bar{a}}^{-1} = \pi_{\bar{a}}^{-1}$. (The second equality is because π_0 is the identity permutation.) Since $\pi_{\bar{a}} = \pi_{a^*}^{-1}$, we get $\pi_{f_{\bar{a},0}(0)} = \pi_{a^*}$. So $f_{\bar{a},0}(0) = a^*$. Since $\bar{c}^{\epsilon}_p = c^{\epsilon}_{f_{\bar{a},0}(p)}$ for all $p \in [n\rangle$, we get $\bar{c}^{\epsilon}_0 = c^{\epsilon}_{f_{\bar{a},0}(0)} = c^{\epsilon}_{a^*}$.

Let $\bar{b} \in [n\rangle$ be the integer such that $f_{\bar{a},0}(\bar{b}) = b^*$. Then $\bar{c}^{\epsilon}_{\bar{b}} = c^{\epsilon}_{f_{\bar{a},0}(\bar{b})} = c^{\epsilon}_{b^*}$. Let $\bar{X} \triangleq \{p \in [n\rangle \mid f_{\bar{a},0}(p) \in X^*\}$. Since $\bar{\mathbf{c}}^{\epsilon}$ is a permutation of the n numbers in $\mathbf{c}^{\epsilon}$ – where $\bar{c}^{\epsilon}_0$ is mapped to $c^{\epsilon}_{a^*}$, $\bar{c}^{\epsilon}_{\bar{b}}$ is mapped to $c^{\epsilon}_{b^*}$, and $\{\bar{c}^{\epsilon}_p | p \in \bar{X}\}$ are mapped to $\{c^{\epsilon}_p | p \in X^*\}$ – and since $\mathbf{c}^{\epsilon} \in \mathcal{C}_{a^*,b^*,X^*,\psi^*}$ with $|X^*| = m - 1$, it is not hard to see (by the definition of $\mathcal{C}_{a,b,X,\psi}$) that there exists a vector $\bar{\psi} \in \{-1, 1\}^m$ such that

$$\bar{\mathbf{c}}^{\epsilon} \in \mathcal{C}_{0,\bar{b},\bar{X},\bar{\psi}} \text{ for all } \epsilon \in (0, \epsilon_0].$$

The rest of the proof is similar to that of Theorem 2. We just need to define $\mathbf{d}^{\epsilon}$ as $\frac{1}{\epsilon g_{i^*,b^*}} \cdot \bar{\mathbf{c}}^{\epsilon}$ instead of $\frac{1}{\epsilon g_{i^*,b^*}} \cdot \mathbf{c}^{\epsilon}$, define $\hat{\mathbf{u}}^{\epsilon}$ as $\frac{1}{\epsilon g_{i^*,b^*}} \cdot \mathbf{u}^{\epsilon}_{\pi_{\bar{a}}}$ instead of $\frac{1}{\epsilon g_{i^*,b^*}} \cdot \mathbf{u}^{\epsilon}$, and then replace (a^*, b^*, X^*, ψ^*) by $(0, \bar{b}, \bar{X}, \bar{\psi})$. In the end, we see that either $z_{0,\bar{b},\bar{X},\bar{\psi}} = \infty$ or $z_{0,\bar{b},\bar{X},-\bar{\psi}} = \infty$. Since here $m = d(\mathcal{C})$, we have $h_m(\mathcal{C}) = \infty = \max_{(0,b,X,\psi)\in\Gamma} z_{0,b,X,\psi}$. □

V. Analog ECCs for One or More Errors

The algorithms for computing the m-heights of Analog ECCs enable us to construct new codes through computer search and optimization. We search for new codes based on Genetic Programming [49]: first, a number of codes are randomly generated (e.g., using Gaussian distributions to randomly generate the generator matrices), their m-heights are computed, and the set of codes with the smallest m-heights are used as the initial population of genetic programming; then, evolutionary operations including mutations (e.g., small perturbations to generator matrices) and crossover (e.g., random combination of two generator matrices) are used to generate new codes, their m-heights are computed, and the set of codes with the smallest m-heights are selected as the new population for the next round of evolutionary operations.

In the above process, when $n = k!$, permutation codes are heavily used for generating the initial population because their m-heights are much faster to compute, which enables the search of more codes for a good initial population. When $n < k!$, we can still let the columns of a generator matrix be permutations of the same set of numbers, and call such codes *Punctured Permutation Codes*. Although the latter codes need to use the m-height algorithm for general codes, experimental results show that they often have good error-correction performance. The details of the code-search algorithm are presented in Algorithm 1.

We present some constructed codes below. The codes can correct one or more errors (i.e., UMEs), and to the best of our knowledge, their error-correction performance (measured by their m-heights) is the best known performance by now.

TABLE III
THE m-HEIGHTS OF DIFFERENT LINEAR $[n, k]$ ANALOG ECCS

$h_m(\mathcal{C})$	$[n,k]=[5,2]$		$[n,k]=[6,2]$		$[n,k]=[6,3]$	
	known	new	known	new	known	new
$m=1$	1		1		1	
$m=2$		1.83	1			2.87
$m=3$		3.25	3	2.28		7
$m=4$	—-	—-		4.10	—-	—-

$h_m(\mathcal{C})$	$[n,k]=[7,2]$		$[n,k]=[7,3]$		$[n,k]=[7,4]$	
	known	new	known	new	known	new
$m=1$	1		1		2	
$m=2$		1	2			3.15
$m=3$		1.90	3			12.56
$m=4$		2.78		10.14	—-	—-
$m=5$		4.95	—-	—-	—-	—-

$h_m(\mathcal{C})$	$[n,k]=[8,2]$		$[n,k]=[8,3]$		$[n,k]=[8,4]$	
	known	new	known	new	known	new
$m=1$	1		1		1	
$m=2$	1			2.49	3	
$m=3$	1			3.71	3	
$m=4$		2.88		9.01		20.96
$m=5$		3.79		20.37	—-	—-
$m=6$		7.16	—-	—-	—-	—-

$h_m(\mathcal{C})$	$[n,k]=[8,5]$		$[n,k]=[9,2]$		$[n,k]=[9,3]$	
	known	new	known	new	known	new
$m=1$	2		1		1	
$m=2$		7.18		1	1	
$m=3$		22.48		1		3.05
$m=4$	—-	—-		1.92		5.76
$m=5$	—-	—-		3.32		12.71
$m=6$	—-	—-		3.88		29.92
$m=7$	—-	—-		12.76	—-	—-

$h_m(\mathcal{C})$	$[n,k]=[9,4]$		$[n,k]=[9,5]$		$[n,k]=[9,6]$	
	known	new	known	new	known	new
$m=1$	1		2		2	
$m=2$		3.38	4			8.56
$m=3$		5.42		12.51		48.72
$m=4$		12.32		76.49	—-	—-
$m=5$		99.90	—-	—-	—-	—-

$h_m(\mathcal{C})$	$[n,k]=[10,2]$		$[n,k]=[10,3]$		$[n,k]=[10,4]$	
	known	new	known	new	known	new
$m=1$	1		1		1	
$m=2$	1			1	≤ 3	
$m=3$	1			2.12		5.00
$m=4$	1			4.36		7.22
$m=5$		1.92		5.97		21.56
$m=6$		3.88		17.18		300.92
$m=7$		3.88		69.44	—-	—-
$m=8$		28.74	—-	—-	—-	—-

$h_m(\mathcal{C})$	$[n,k]=[10,5]$		$[n,k]=[10,6]$		$[n,k]=[10,7]$	
	known	new	known	new	known	new
$m=1$	1		2		3	
$m=2$		4.23	4			12.86
$m=3$		8.90		23.06		89.81
$m=4$		29.80		273.24	—-	—-
$m=5$		334.75	—-	—-	—-	—-

$$G^2_{5,2} = \begin{bmatrix} 0.911 & 0.03 & 1.481 & -0.756 & 1.249 \\ -0.049 & 0.975 & 1.511 & -1.303 & 0.74 \end{bmatrix}$$

$$G^3_{5,2} = \begin{bmatrix} 0.909 & 0.014 & 1.493 & -0.738 & 1.235 \\ -0.039 & 0.967 & 1.519 & -1.299 & 0.719 \end{bmatrix}$$

$$G^3_{6,2} = \begin{bmatrix} 0.859 & -0.103 & 0.746 & -0.35 & 0.892 & -0.95 \\ 0.025 & 1.063 & -1.591 & -0.721 & -1.115 & 0.679 \end{bmatrix}$$

$$G^4_{6,2} = \begin{bmatrix} 0.922 & -0.121 & 0.746 & -0.368 & 0.896 & -1.002 \\ 0.022 & 1.069 & -1.593 & -0.702 & -1.089 & 0.676 \end{bmatrix}$$

$$G^2_{6,3} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.7 & -0.791 & 0.043 \\ 0.0 & 1.0 & 0.0 & -0.647 & 0.833 & 1.162 \\ 0.0 & 0.0 & 1.0 & 0.311 & 1.284 & 1.028 \end{bmatrix}$$

$$G^3_{6,3} = \begin{bmatrix} 0.236 & -1.136 & -0.45 & -1.136 & 0.236 & -0.45 \\ -1.136 & -0.45 & 0.236 & 0.236 & -0.45 & -1.136 \\ -0.45 & 0.236 & -1.136 & -0.45 & -1.136 & 0.236 \end{bmatrix}$$

$$G^3_{7,2} = \begin{bmatrix} 1.0 & 0.0 & -0.74 & -1.09 & 0.463 & 0.584 & 0.719 \\ 0.0 & 1.0 & 1.106 & 0.79 & 0.713 & 0.706 & -0.531 \end{bmatrix}$$

$$G^4_{7,2} = \begin{bmatrix} 1.189 & 0.182 & 0.503 & -1.479 & 0.919 & 1.385 & 1.155 \\ 0.016 & 1.092 & -1.447 & -0.772 & 1.194 & -0.551 & -1.169 \end{bmatrix}$$

$$G^5_{7,2} = \begin{bmatrix} 1.167 & 0.206 & 0.483 & -1.49 & 0.939 & 1.344 & 1.162 \\ 0.028 & 1.129 & -1.412 & -0.753 & 1.173 & -0.565 & -1.198 \end{bmatrix}$$

$$G^4_{7,3} = \begin{bmatrix} 1.087 & -0.085 & -0.087 & 1.134 & 0.5 & 0.19 & 0.955 \\ -0.037 & 0.977 & -0.033 & -0.323 & -0.315 & 0.635 & 0.394 \\ -0.109 & 0.035 & 1.039 & 0.527 & -1.093 & 0.751 & 0.159 \end{bmatrix}$$

$$G^2_{7,4} = \begin{bmatrix} -0.822 & -1.211 & -1.211 & -0.816 & -0.816 & -0.405 & -0.816 \\ -0.816 & -0.822 & -0.405 & -1.211 & -0.405 & -0.816 & -0.822 \\ -1.211 & -0.405 & -0.816 & -0.822 & -0.822 & -1.211 & -0.405 \\ -0.405 & -0.816 & -0.822 & -0.405 & -1.211 & -0.822 & -1.211 \end{bmatrix}$$

$$G^3_{7,4} = \begin{bmatrix} 0.976 & -0.055 & 0.059 & -0.001 & 0.842 & -0.996 & 0.804 \\ 0.001 & 1.047 & -0.035 & -0.057 & -0.632 & 0.481 & 0.336 \\ 0.015 & -0.014 & 0.954 & 0.015 & -0.686 & 1.743 & -0.433 \\ -0.042 & -0.111 & -0.076 & 1.016 & -0.172 & -0.524 & -0.939 \end{bmatrix}$$

$$G^4_{8,2} = \begin{bmatrix} 0.962 & 0.051 & 0.765 & -0.947 & 0.629 & -0.254 & 1.113 & -0.283 \\ -0.003 & 1.05 & 0.291 & -0.938 & 0.878 & 0.286 & 0.79 & -0.753 \end{bmatrix}$$

$$G^5_{8,2} = \begin{bmatrix} 1.0 & 0.0 & 0.534 & 0.583 & -0.709 & -0.053 & 0.543 & 0.927 \\ 0.0 & 1.0 & 0.611 & -1.211 & 0.233 & -0.784 & -0.659 & -0.947 \end{bmatrix}$$

$$G^6_{8,2} = \begin{bmatrix} 0.967 & 0.049 & 0.789 & -0.956 & 0.619 & -0.206 & 1.107 & -0.276 \\ -0.004 & 1.001 & 0.273 & -0.935 & 0.859 & 0.308 & 0.79 & -0.736 \end{bmatrix}$$

$$G^2_{8,3} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & -0.254 & 0.511 & -0.749 & -0.73 & -0.73 \\ 0.0 & 1.0 & 0.0 & -1.536 & -0.006 & 0.174 & 1.339 & 1.339 \\ 0.0 & 0.0 & 1.0 & -0.624 & 0.705 & 1.01 & 0.943 & 0.943 \end{bmatrix}$$

Fig. 2. Generator matrices of some new Analog ECCs in Table III, from $n = 5$, $k = 2$, $m = 2$ to $n = 8$, $k = 3$, $m = 2$.

$$G^3_{8,3} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 1.157 & 0.017 & 1.164 & 1.353 & 1.353 \\ 0.0 & 1.0 & 0.0 & -0.076 & -0.741 & -0.888 & -1.495 & -1.495 \\ 0.0 & 0.0 & 1.0 & 1.075 & -0.702 & 0.876 & -0.266 & -0.266 \end{bmatrix}$$

$$G^4_{8,3} = \begin{bmatrix} 1.049 & 0.026 & 0.021 & 0.893 & 0.809 & 0.27 & -1.151 & 1.402 \\ -0.008 & 1.112 & 0.022 & 0.52 & -1.689 & -2.2 & 0.285 & 0.396 \\ -0.076 & 0.101 & 0.996 & -0.529 & -1.262 & 1.039 & -0.979 & 0.582 \end{bmatrix}$$

$$G^5_{8,3} = \begin{bmatrix} 1.042 & 0.06 & 0.036 & 0.887 & 0.807 & 0.269 & -1.179 & 1.419 \\ 0.006 & 1.086 & 0.036 & 0.506 & -1.693 & -2.209 & 0.303 & 0.394 \\ -0.076 & 0.092 & 0.987 & -0.541 & -1.271 & 1.042 & -0.99 & 0.574 \end{bmatrix}$$

$$G^4_{8,4} = \begin{bmatrix} 0.998 & 0.137 & 0.088 & 0.02 & 1.116 & 0.578 & 0.909 & -0.369 \\ 0.005 & 1.104 & 0.088 & -0.048 & -0.461 & 1.039 & -1.613 & 1.124 \\ -0.057 & 0.147 & 1.019 & -0.021 & -1.488 & 0.924 & 0.445 & 0.438 \\ 0.021 & 0.115 & 0.032 & 0.993 & -1.284 & 1.769 & 0.214 & -0.831 \end{bmatrix}$$

$$G^2_{8,5} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.756 & -1.029 & -0.854 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 1.077 & 0.788 & -1.196 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.881 & -1.261 & 0.173 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & -0.629 & -0.715 & 1.49 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.66 & 1.036 & 0.018 \end{bmatrix}$$

$$G^3_{8,5} = \begin{bmatrix} 0.9 & -0.235 & 0.009 & -0.051 & -0.187 & 1.141 & 1.339 & -1.486 \\ 0.308 & 0.975 & 0.03 & -0.087 & 0.167 & 0.158 & -0.36 & 0.938 \\ -0.068 & -0.067 & 0.87 & 0.273 & 0.22 & -2.267 & -0.768 & -0.079 \\ -0.001 & -0.047 & 0.096 & 0.939 & 0.159 & -1.386 & -0.706 & 0.862 \\ -0.052 & -0.222 & -0.211 & -0.031 & 1.266 & 1.024 & -1.253 & 0.901 \end{bmatrix}$$

$$G^4_{9,2} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.476 & 0.476 & 0.994 & 0.994 & 0.819 & 0.819 \\ 0.0 & 1.0 & 1.0 & -0.896 & -0.896 & -0.744 & -0.744 & 0.335 & 0.335 \end{bmatrix}$$

$$G^5_{9,2} = \begin{bmatrix} 1.0 & 0.0 & 0.865 & 0.111 & -1.464 & 1.522 & 0.364 & 0.511 & -1.228 \\ 0.0 & 1.0 & 0.168 & 0.916 & 0.509 & -0.935 & 0.776 & -0.863 & 1.517 \end{bmatrix}$$

$$G^6_{9,2} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 1.586 & 1.586 & -0.655 & -0.655 & 1.193 & 1.193 \\ 0.0 & 1.0 & 1.0 & 1.555 & 1.555 & -1.244 & -1.244 & 0.684 & 0.684 \end{bmatrix}$$

$$G^7_{9,2} = \begin{bmatrix} 1.004 & 0.017 & -1.55 & 0.789 & -0.357 & 1.22 & 0.243 & 0.637 & 0.533 \\ -0.002 & 0.948 & -0.258 & 0.601 & 0.525 & -0.636 & -0.957 & -0.182 & -0.414 \end{bmatrix}$$

$$G^3_{9,3} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & -0.298 & -0.954 & 1.01 & -0.104 & -0.145 & 0.893 \\ 0.0 & 1.0 & 0.0 & 0.004 & 0.784 & -1.751 & -0.882 & -0.802 & -1.202 \\ 0.0 & 0.0 & 1.0 & -0.911 & 0.346 & 0.841 & 0.655 & 0.613 & -0.594 \end{bmatrix}$$

$$G^4_{9,3} = \begin{bmatrix} 1.003 & 0.003 & -0.011 & -0.073 & -0.461 & 0.676 & 0.485 & 0.631 & -0.72 \\ 0.021 & 0.999 & 0.018 & 0.269 & -0.649 & 0.202 & -0.858 & -0.421 & 1.135 \\ 0.013 & 0.003 & 1.009 & -0.764 & 1.034 & 1.532 & -0.095 & -0.384 & -0.392 \end{bmatrix}$$

$$G^5_{9,3} = \begin{bmatrix} 0.992 & -0.02 & 0.009 & -1.579 & -0.204 & -0.952 & 0.121 & 0.519 & -1.571 \\ -0.014 & 0.989 & 0.005 & 1.215 & -1.134 & -0.922 & -0.475 & -0.331 & 0.192 \\ -0.004 & 0.005 & 0.98 & 0.052 & 1.081 & -0.457 & 0.69 & -0.476 & -0.885 \end{bmatrix}$$

$$G^6_{9,3} = \begin{bmatrix} 0.992 & -0.026 & -0.019 & -0.08 & -0.487 & 0.645 & 0.473 & 0.598 & -0.732 \\ 0.002 & 0.969 & 0.032 & 0.291 & -0.671 & 0.208 & -0.837 & -0.398 & 1.118 \\ -0.006 & -0.013 & 1.014 & -0.805 & 1.011 & 1.518 & -0.114 & -0.347 & -0.408 \end{bmatrix}$$

Fig. 3. Generator matrices of some new Analog ECCs in Table III, from $n = 8$, $k = 3$, $m = 3$ to $n = 9$, $k = 3$, $m = 6$.

A summary of the codes is shown in Table III. The table lists the m-heights of different linear $[n, k]$ Analog ECCs. Here the columns labelled by "known" refer to known codes from [1], and the columns labelled by "new" refer to the new codes presented here. [3] Notice that when $m \geq 4$, codes with finite m-heights can correct two or more UMEs; and the smaller the m-height is, the smaller the ratio Δ/δ (where Δ and δ are the two threshold values for UMEs and LMEs as described earlier) needs to be. Many of the new codes here can correct two or more errors. Some of them have finite m-heights for m as large as 8 (e.g., the $[10, 2]$ code in the table), which means they can correct up to 4 UMEs.

Some codes in Table III have m-height equal to 1 for some m values. They are either the Repetition Code or the Cartesian

[3] Only m-heights of finite values are shown in the table. And for those cells in the table where $m > n-k$, we cross them out by a horizontal line because by the Singleton bound, the minimum distance of the code $d(\mathcal{C}) \leq n-k+1$, so $h_m(\mathcal{C}) = \infty$ when $m > n-k$ for any code $\mathcal{C}$.

$$G^2_{9,4} = \begin{bmatrix} 1.386 & -0.623 & -0.658 & -0.658 & -0.623 & -0.623 & 1.772 & 1.386 & -0.658 \\ -0.623 & 1.386 & -0.623 & 1.772 & 1.772 & -0.658 & 1.386 & -0.623 & 1.386 \\ -0.658 & -0.658 & 1.386 & 1.386 & -0.658 & 1.772 & -0.658 & 1.772 & 1.772 \\ 1.772 & 1.772 & 1.772 & -0.623 & 1.386 & 1.386 & -0.623 & -0.658 & -0.623 \end{bmatrix}$$

$$G^3_{9,4} = \begin{bmatrix} -1.838 & 1.765 & -1.838 & 0.145 & 0.145 & 1.765 & 1.765 & -1.838 & -0.641 \\ -0.641 & 0.145 & 1.765 & -0.641 & -1.838 & 0.145 & -1.838 & 0.145 & -1.838 \\ 0.145 & -0.641 & 0.145 & -1.838 & -0.641 & -1.838 & 0.145 & 1.765 & 1.765 \\ 1.765 & -1.838 & -0.641 & 1.765 & 1.765 & -0.641 & -0.641 & -0.641 & 0.145 \end{bmatrix}$$

$$G^4_{9,4} = \begin{bmatrix} 0.099 & 0.099 & -1.054 & 0.099 & -1.054 & -1.549 & -1.054 & -0.514 & 0.099 \\ -0.514 & -1.549 & -0.514 & -0.514 & -1.549 & -1.054 & -0.514 & 0.099 & -1.054 \\ -1.054 & -1.054 & -1.549 & -1.549 & 0.099 & 0.099 & 0.099 & -1.549 & -0.514 \\ -1.549 & -0.514 & 0.099 & -1.054 & -0.514 & -0.514 & -1.549 & -1.054 & -1.549 \end{bmatrix}$$

$$G^5_{9,4} = \begin{bmatrix} 0.994 & -0.014 & 0.006 & 0.024 & -0.32 & 0.427 & -0.414 & -0.897 & 0.429 \\ -0.02 & 1.022 & -0.006 & 0.02 & 0.649 & -0.593 & 0.469 & 0.595 & 0.16 \\ -0.006 & -0.035 & 1.017 & 0.019 & -0.233 & -1.258 & 0.611 & 0.422 & 0.094 \\ 0.023 & 0.013 & 0.011 & 1.028 & 0.45 & 0.82 & -1.787 & -1.595 & 0.438 \end{bmatrix}$$

$$G^3_{9,5} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & -0.761 & 0.46 & -0.524 & 0.79 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.487 & -0.86 & 1.421 & -0.091 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & -0.789 & 0.552 & -1.035 & -0.935 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & -0.353 & -0.764 & 0.075 & -0.781 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & -0.147 & -1.116 & 0.873 & -0.328 \end{bmatrix}$$

$$G^4_{9,5} = \begin{bmatrix} 0.999 & 0.559 & 2.218 & -0.129 & 0.551 & 0.57 & 0.989 & 1.411 & 0.987 \\ 2.207 & 1.45 & -0.161 & 1.379 & 1.476 & 2.237 & 0.562 & -0.152 & 1.419 \\ 1.449 & 0.994 & 0.983 & 2.181 & -0.167 & 0.971 & 1.462 & 0.545 & 2.232 \\ -0.209 & 2.273 & 1.478 & 0.489 & 2.23 & -0.173 & 2.241 & 0.984 & 0.569 \\ 0.551 & -0.145 & 0.545 & 1.015 & 0.988 & 1.458 & -0.112 & 2.236 & -0.132 \end{bmatrix}$$

$$G^2_{9,6} = \begin{bmatrix} -0.553 & -0.817 & 0.513 & -0.784 & -0.215 & -0.784 & -0.784 & -0.215 & 0.513 \\ -0.215 & -0.215 & -0.553 & -0.553 & 0.513 & 0.513 & -0.215 & -0.553 & -0.215 \\ -0.817 & -0.784 & 0.293 & -0.215 & -0.817 & -0.553 & 0.513 & 0.513 & 0.293 \\ 0.513 & -0.553 & -0.817 & 0.513 & 0.293 & -0.215 & -0.817 & 0.293 & -0.784 \\ -0.784 & 0.513 & -0.215 & -0.817 & -0.784 & 0.293 & 0.293 & -0.784 & -0.553 \\ 0.293 & 0.293 & -0.784 & 0.293 & -0.553 & -0.817 & -0.553 & -0.817 & -0.817 \end{bmatrix}$$

$$G^3_{9,6} = \begin{bmatrix} 0.212 & 0.271 & 0.38 & -1.832 & -0.588 & -0.479 & -1.848 & -0.435 & 1.37 \\ -0.409 & -1.821 & 0.171 & -0.431 & 0.33 & -1.863 & -0.5 & 0.179 & -0.532 \\ -0.553 & 0.238 & -0.402 & 1.345 & -1.727 & 1.27 & 0.328 & 1.335 & 0.334 \\ 1.19 & -0.525 & 1.31 & -0.58 & 0.16 & 0.42 & 1.296 & 0.333 & -1.84 \\ -1.836 & 1.296 & -1.833 & 0.185 & 1.388 & -0.616 & -0.523 & -1.823 & 0.206 \\ 0.319 & -0.433 & -0.571 & 0.319 & -0.437 & 0.251 & 0.24 & -0.428 & -0.531 \end{bmatrix}$$

$$G^5_{10,2} = \begin{bmatrix} 1.0 & 1.0 & 0.0 & 0.0 & 0.476 & 0.476 & 0.994 & 0.994 & 0.819 & 0.819 \\ 0.0 & 0.0 & 1.0 & 1.0 & -0.896 & -0.896 & -0.744 & -0.744 & 0.335 & 0.335 \end{bmatrix}$$

$$G^{6,7}_{10,2} = \begin{bmatrix} 1.0 & 1.0 & 0.0 & 0.0 & 1.586 & 1.586 & -0.655 & -0.655 & 1.193 & 1.193 \\ 0.0 & 0.0 & 1.0 & 1.0 & 1.555 & 1.555 & -1.244 & -1.244 & 0.684 & 0.684 \end{bmatrix}$$

$$G^8_{10,2} = \begin{bmatrix} 1.009 & 0.034 & -1.678 & -0.319 & -0.09 & 0.407 & 0.235 & -0.77 & -1.429 & 1.751 \\ -0.036 & 0.956 & -0.62 & -0.721 & 1.066 & -1.115 & 1.612 & -0.616 & -0.221 & -0.402 \end{bmatrix}$$

$$G^3_{10,3} = \begin{bmatrix} 1.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.228 & -1.228 & -0.087 & -0.087 \\ 0.0 & 0.0 & 1.0 & 1.0 & 0.0 & 0.0 & -0.787 & -0.787 & 1.041 & 1.041 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 1.0 & 0.442 & 0.442 & -0.991 & -0.991 \end{bmatrix}$$

$$G^4_{10,3} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 1.773 & -1.133 & 1.017 & -1.559 & 0.195 & -1.01 & -0.374 \\ 0.0 & 1.0 & 0.0 & 1.303 & 0.257 & 0.401 & -0.526 & -1.155 & 0.397 & 0.79 \\ 0.0 & 0.0 & 1.0 & 0.305 & 0.283 & -0.125 & -0.715 & -0.971 & -0.426 & -0.885 \end{bmatrix}$$

Fig. 4. Generator matrices of some new Analog ECCs in Table III, from $n = 9$, $k = 4$, $m = 2$ to $n = 10$, $k = 3$, $m = 4$.

$$G^5_{10,3} = \begin{bmatrix} 1.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & -0.761 & -0.761 & -1.85 & -1.85 \\ 0.0 & 0.0 & 1.0 & 1.0 & 0.0 & 0.0 & -0.665 & -0.665 & 0.715 & 0.715 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 1.0 & -0.748 & -0.748 & -0.609 & -0.609 \end{bmatrix}$$

$$G^6_{10,3} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 1.773 & -1.133 & 1.017 & -1.559 & 0.195 & -1.01 & -0.374 \\ 0.0 & 1.0 & 0.0 & 1.303 & 0.257 & 0.401 & -0.526 & -1.155 & 0.397 & 0.79 \\ 0.0 & 0.0 & 1.0 & 0.305 & 0.283 & -0.125 & -0.715 & -0.971 & -0.426 & -0.885 \end{bmatrix}$$

$$G^7_{10,3} = \begin{bmatrix} 1.004 & -0.001 & -0.003 & 0.893 & 1.511 & -0.375 & 0.208 & -1.17 & -1.427 & 0.057 \\ 0.005 & 1.002 & 0.005 & -0.218 & -0.053 & 0.616 & 0.583 & -0.312 & -0.901 & -1.056 \\ 0.0 & 0.007 & 1.023 & 0.239 & 2.255 & 0.317 & 0.838 & 0.109 & 2.005 & -0.284 \end{bmatrix}$$

$$G^3_{10,4} = \begin{bmatrix} 0.99 & -0.656 & 0.982 & -0.242 & 1.936 & 0.974 & 0.907 & -0.394 & -1.116 & -0.405 \\ -1.11 & -1.011 & 1.886 & 0.909 & 0.981 & -0.382 & -0.507 & -1.077 & -0.525 & 0.967 \\ -0.292 & 1.128 & -0.489 & -1.184 & -1.206 & -1.137 & 2.037 & 2.001 & 1.072 & 1.879 \\ 2.062 & 1.953 & -1.101 & 2.014 & -0.506 & 1.968 & -1.178 & 0.858 & 1.994 & -1.033 \end{bmatrix}$$

$$G^4_{10,4} = \begin{bmatrix} -0.513 & 0.43 & -1.828 & -1.828 & -0.513 & -1.254 & -1.828 & -1.254 & -0.513 & 0.43 \\ 0.43 & -1.828 & -0.513 & 0.43 & -1.828 & -1.828 & -0.513 & -1.828 & -1.828 & -0.513 \\ -1.254 & -1.254 & -1.254 & -0.513 & -1.254 & -0.513 & 0.43 & 0.43 & 0.43 & -1.828 \\ -1.828 & -0.513 & 0.43 & -1.254 & 0.43 & 0.43 & -1.254 & -0.513 & -1.254 & -1.254 \end{bmatrix}$$

$$G^5_{10,4} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.026 & -0.61 & 1.582 & -0.327 & -0.91 & -0.927 \\ 0.0 & 1.0 & 0.0 & 0.0 & 2.279 & 0.053 & 1.797 & 0.435 & -0.994 & 0.822 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.342 & -0.774 & 0.264 & -0.07 & -0.617 & -0.853 \\ 0.0 & 0.0 & 0.0 & 1.0 & -1.356 & -1.203 & 1.312 & -1.275 & -0.132 & -0.236 \end{bmatrix}$$

$$G^6_{10,4} = \begin{bmatrix} 1.016 & -0.021 & -0.006 & -0.008 & 1.22 & 0.271 & 0.06 & -0.756 & -1.301 & -1.257 \\ -0.008 & 1.001 & -0.001 & 0.021 & -1.037 & 0.024 & -0.321 & 1.061 & 0.412 & -0.362 \\ -0.006 & 0.013 & 0.988 & -0.015 & 0.452 & -0.168 & -0.781 & 0.677 & -0.189 & -0.171 \\ -0.03 & 0.022 & -0.008 & 0.986 & 0.007 & 2.065 & -2.632 & 0.765 & 0.137 & 0.908 \end{bmatrix}$$

$$G^2_{10,5} = \begin{bmatrix} 0.005 & 0.992 & 0.005 & -1.276 & 0.005 & 1.974 & 1.974 & -1.276 & -1.276 & -0.033 \\ 1.974 & 1.974 & -0.033 & 0.005 & -0.033 & -0.033 & -1.276 & 0.992 & 0.992 & -1.276 \\ -0.033 & -1.276 & 1.974 & 0.992 & 1.974 & 0.992 & 0.992 & 1.974 & 0.005 & 1.974 \\ 0.992 & 0.005 & 0.992 & -0.033 & -1.276 & -1.276 & -0.033 & -0.033 & 1.974 & 0.992 \\ -1.276 & -0.033 & -1.276 & 1.974 & 0.992 & 0.005 & 0.005 & 0.005 & -0.033 & 0.005 \end{bmatrix}$$

$$G^3_{10,5} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & -0.027 & 1.437 & 1.213 & -0.215 & -0.605 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & -0.498 & 0.603 & -0.112 & 1.005 & -0.565 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 1.341 & -1.72 & -0.216 & -1.118 & -0.757 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.828 & 0.354 & -1.654 & 0.634 & -0.592 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 1.928 & -1.663 & 0.434 & 0.421 & -0.812 \end{bmatrix}$$

$$G^4_{10,5} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.481 & -0.917 & -1.054 & -1.92 & 0.053 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & -0.131 & 1.088 & 1.252 & 0.611 & 1.447 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & -0.664 & -0.214 & -0.6 & -1.374 & 0.131 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & -1.139 & -1.8 & 1.008 & -1.216 & 0.782 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & -0.693 & 0.812 & 0.957 & 0.895 & 1.457 \end{bmatrix}$$

$$G^5_{10,5} = \begin{bmatrix} 1.019 & -0.005 & 0.004 & 0.0 & 0.011 & -0.284 & -0.257 & 1.023 & 1.625 & -1.34 \\ -0.002 & 1.007 & -0.005 & -0.012 & 0.002 & 0.542 & -1.197 & 0.834 & 0.799 & -0.305 \\ 0.002 & 0.017 & 1.016 & -0.019 & 0.009 & -0.13 & 0.339 & 1.248 & -0.916 & -0.063 \\ -0.001 & -0.008 & 0.015 & 1.001 & 0.008 & 0.975 & -1.226 & -0.841 & 0.916 & 0.26 \\ 0.005 & 0.007 & 0.012 & 0.004 & 1.005 & 2.346 & -1.832 & -0.194 & -1.172 & 0.64 \end{bmatrix}$$

Fig. 5. Generator matrices of some new Analog ECCs in Table III, from $n = 10$, $k = 3$, $m = 5$ to $n = 10$, $k = 5$, $m = 5$.

Power of Repetition Code presented in [1], or their simple extension: for an $[n, k]$ code, repeat each of the k information numbers either $\lceil n/k \rceil$ or $\lfloor n/k \rfloor$ times in the codeword. We call such codes *Generalized Repetition Codes*.

The generator matrices of the new codes in Table III are shown in Fig. 2 through Fig. 6, except those of Generalized Repetition Codes due to their simplicity. Each $G^m_{n,k}$ in the figures refers to the generator matrix of the new $[n, k]$ code whose m-height is shown in Table III. (The matrix $G^{6,7}_{10,2}$ in Fig. 4 is the generator matrix of the $[10, 2]$ code for both $m = 6$ and $m = 7$.)

It is worthwhile to note that extending an $[n, k]$ Analog ECC to an $[n + 1, k]$ code by adding a coordinate does not always decrease the m-height [1]. (Sometimes it increases it.) That makes the code design problem different from conventional ECCs, where the error-correction capability is determined by the minimum distance $d(\mathcal{C})$ (instead of $h_m(\mathcal{C})$), and therefore extending a code can only increase that capability. We also note that some of the new codes in Table III are Analog Permutation Codes or Punctured Permutation Codes, including the codes with generator matrices $G^3_{6,3}$ and $G^2_{7,4}$ in Fig. 2, $G^2_{9,4}$, $G^3_{9,4}$, $G^4_{9,4}$ and $G^2_{9,6}$ in Fig. 4, $G^4_{10,4}$ and $G^2_{10,5}$ in Fig. 5,

$$G^3_{10,6} = \begin{bmatrix} 1.043 & 0.569 & 0.569 & 0.023 & 0.023 & 1.196 & 0.569 & 1.043 & -0.201 & 1.196 \\ -0.201 & -0.201 & -0.201 & -0.201 & 1.196 & -1.141 & -1.141 & 1.196 & 1.043 & -0.201 \\ 1.196 & -1.141 & -1.141 & 1.043 & 0.569 & 0.023 & -0.201 & -1.141 & 1.196 & 0.023 \\ 0.023 & 1.196 & 0.023 & 1.196 & -0.201 & 1.043 & 1.196 & 0.023 & 0.569 & -1.141 \\ 0.569 & 1.043 & 1.043 & -1.141 & -1.141 & -0.201 & 0.023 & 0.569 & 0.023 & 1.043 \\ -1.141 & 0.023 & 1.196 & 0.569 & 1.043 & 0.569 & 1.043 & -0.201 & -1.141 & 0.569 \end{bmatrix}$$

$$G^2_{10,7} = \begin{bmatrix} 0.556 & 0.96 & 0.556 & 0.63 & 0.556 & 0.556 & -0.507 & -0.507 & 0.319 & 0.319 \\ 0.63 & 0.491 & -0.58 & 0.96 & 0.63 & -0.507 & 0.63 & 0.319 & 0.556 & 0.491 \\ 0.319 & -0.58 & 0.319 & -0.507 & 0.96 & 0.491 & 0.319 & 0.63 & -0.507 & 0.96 \\ -0.507 & -0.507 & -0.507 & 0.491 & 0.491 & 0.319 & 0.491 & 0.491 & 0.491 & -0.507 \\ -0.58 & 0.556 & 0.491 & 0.319 & 0.319 & 0.96 & 0.96 & -0.58 & 0.63 & 0.556 \\ 0.96 & 0.63 & 0.96 & -0.58 & -0.507 & 0.63 & -0.58 & 0.96 & -0.58 & -0.58 \\ 0.491 & 0.319 & 0.63 & 0.556 & -0.58 & -0.58 & 0.556 & 0.556 & 0.96 & 0.63 \end{bmatrix}$$

$$G^4_{10,6} = \begin{bmatrix} -0.375 & 0.401 & -0.246 & -0.233 & -0.355 & -0.385 & 0.472 & 0.401 & -0.389 & 0.128 \\ 0.793 & 0.791 & 0.114 & -0.378 & 0.407 & 0.801 & -0.252 & 0.808 & 0.801 & -0.256 \\ 0.418 & -0.262 & 0.783 & 0.482 & 0.103 & 0.401 & 0.784 & 0.49 & 0.48 & 0.783 \\ 0.48 & 0.111 & -0.386 & 0.785 & -0.24 & 0.114 & 0.403 & -0.374 & -0.272 & 0.491 \\ -0.249 & -0.356 & 0.408 & 0.119 & 0.791 & 0.479 & 0.126 & -0.257 & 0.116 & -0.363 \\ 0.116 & 0.485 & 0.476 & 0.403 & 0.501 & -0.257 & -0.385 & 0.123 & 0.414 & 0.381 \end{bmatrix}$$

$$G^3_{10,7} = \begin{bmatrix} 1.219 & -0.028 & 1.206 & 0.604 & 0.055 & -2.649 & 1.212 & -2.646 & -0.373 & 0.004 \\ 0.04 & 1.183 & -2.666 & -0.387 & -0.033 & 0.07 & -0.863 & 0.044 & 0.63 & -2.537 \\ 0.622 & 0.039 & -0.075 & 0.123 & 0.62 & 0.663 & -0.06 & -0.825 & 0.112 & 1.151 \\ -0.815 & -0.369 & 0.065 & -0.045 & -0.345 & -0.831 & -2.596 & -0.421 & -2.608 & -0.106 \\ -0.09 & -0.813 & -0.38 & -0.838 & -2.602 & 1.133 & 0.053 & 0.008 & 1.203 & -0.794 \\ -2.583 & 0.672 & -0.864 & -2.654 & -0.834 & -0.41 & -0.385 & 1.166 & -0.113 & 0.584 \\ -0.411 & -2.657 & 0.648 & 1.183 & 1.2 & -0.084 & 0.638 & 0.543 & -0.824 & -0.364 \end{bmatrix}$$

Fig. 6. Generator matrices of some new Analog ECCs in Table III, from $n = 10$, $k = 6$, $m = 3$ to $n = 10$, $k = 7$, $m = 3$.

and $G^3_{10,6}$ and $G^2_{10,7}$ in Fig. 6. The new codes in Table III, with their relatively small m-heights for multiple values of m – and therefore the corresponding error correction capabilities – that were not known before, can form a basis for the search and construction of more codes in the future.

VI. Conclusion

Analog ECCs consider small ubiquitous noise as tolerable, and focus on correcting errors of larger magnitudes that can significantly affect the performance of machine learning algorithms. One application is the implementation of deep neural networks in nanoscale analog circuits (a realization of in-memory computing), where Analog ECCs can make the widely used vector-matrix multiplication operations more reliable. In this paper, algorithms are presented for computing the m-heights of Analog ECCs, which are directly related to the codes' error-correction capabilities. Codes with special structures that can further reduce the complexity of the m-height algorithm, namely Analog Permutation Codes, are also presented. The algorithms are used to find the exact m-heights of known codes, and for finding more codes that correct one or more errors. A list of new codes with the best known performance are summarized.

The study on Analog ECCs can be extended in multiple ways. One important extension is to consider quantization errors in analog systems, study their effect on error correction, and optimize Analog ECCs accordingly. Another important topic is to design efficient decoding algorithms that are easily implementable in analog circuits, and adapt the codes to the specific requirements of the analog system. Those topics remain as our future research directions.

References

[1] R. M. Roth, "Analog error-correcting codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 7, pp. 4075–4088, Jul. 2020.

[2] R. M. Roth, "Fault-tolerant neuromorphic computing on nanoscale crossbar architectures," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2022, pp. 202–207.

[3] M. Le Gallo et al., "A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference," *Nature Electron.*, vol. 6, no. 9, pp. 680–693, Aug. 2023.

[4] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnol.*, vol. 15, no. 7, pp. 529–544, Jul. 2020.

[5] W. Wang et al., "A memristive deep belief neural network based on silicon synapses," *Nature Electron.*, vol. 5, no. 12, pp. 870–880, Dec. 2022.

[6] W. Zhang et al., "Edge learning using a fully integrated neuro-inspired memristor chip," *Science*, vol. 381, no. 6663, pp. 1205–1211, Sep. 2023.

[7] H.-S.-P. Wong and S. Salahuddin, "Memory leads the way to better computing," *Nature Nanotechnol.*, vol. 10, no. 3, pp. 191–194, Mar. 2015.

[8] P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, Jan. 2020.

[9] C.-X. Xue et al., "A CMOS-integrated compute-in-memory macro based on resistive random-access memory for AI edge devices," *Nature Electron.*, vol. 4, no. 1, pp. 81–90, Dec. 2020.

[10] C. Li, R. M. Roth, C. Graves, X. Sheng, and J. P. Strachan, "Analog error correcting codes for defect tolerant matrix multiplication in crossbars," in *IEDM Tech. Dig.*, Dec. 2020, pp. 36.6.1–36.6.4.

[11] K. Huang, P. H. Siegel, and A. Jiang, "Functional error correction for robust neural networks," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 267–276, May 2020.

[12] R. M. Roth, "Fault-tolerant dot-product engines," *IEEE Trans. Inf. Theory*, vol. 65, no. 4, pp. 2046–2057, Apr. 2019.

[13] G. O. H. Katona and Á. Seress, "Greedy construction of nearly regular graphs," *Eur. J. Combinatorics*, vol. 14, no. 3, pp. 213–229, May 1993.

[14] B. Chen and G. W. Wornell, "Analog error-correcting codes based on chaotic dynamical systems," *IEEE Trans. Commun.*, vol. 46, no. 7, pp. 881–890, Jul. 1998.

[15] C. E. Shannon, "Communication in the presence of noise," *Proc. IRE*, vol. 37, no. 1, pp. 10–21, Jan. 1949.

[16] V. A. Kotelnikov, *The Theory of Optimum Noise Immunity*. New York, NY, USA: McGraw-Hill, 1959.

[17] Y. Hu, J. Garcia-Frias, and M. Lamarca, "Analog joint source-channel coding using non-linear curves and MMSE decoding," *IEEE Trans. Commun.*, vol. 59, no. 11, pp. 3016–3026, Nov. 2011.

[18] E. Akyol, K. B. Viswanatha, K. Rose, and T. A. Ramstad, "On zero-delay source-channel coding," *IEEE Trans. Inf. Theory*, vol. 60, no. 12, pp. 7473–7489, Dec. 2014.

[19] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, vol. 89, 2019, pp. 1215–1225.

[20] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog Lagrange coded computing," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 283–295, Mar. 2021.

[21] N. Charalambides, H. Mahdavifar, and A. O. Hero, "Numerically stable binary gradient coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 2622–2627.

[22] A. B. Das and A. Ramamoorthy, "Distributed matrix-vector multiplication: A convolutional coding approach," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 3022–3026.

[23] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 3017–3021.

[24] M. V. Jamali, M. Soleymani, and H. Mahdavifar, "Coded distributed computing: Performance limits and code designs," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Aug. 2019, pp. 1–5.

[25] A. Ramamoorthy and L. Tang, "Numerically stable coded matrix computations via circulant and rotation matrix embeddings," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 1712–1717.

[26] S. Arora and S. Khot, "Fitting algebraic curves to noisy data," *J. Comput. Syst. Sci.*, vol. 67, no. 2, pp. 325–340, Sep. 2003.

[27] V. Guruswami and D. Zuckerman, "Robust Fourier and polynomial curve fitting," in *Proc. IEEE 57th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2016, pp. 751–759.

[28] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error characterization, mitigation, and recovery in flash-memory-based solid-state drives," *Proc. IEEE*, vol. 105, no. 9, pp. 1666–1704, Sep. 2017.

[29] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited-magnitude errors with application to multilevel flash memories," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1582–1595, Apr. 2010.

[30] N. Elarief and B. Bose, "Optimal, systematic, q-ary codes correcting all asymmetric and symmetric errors of limited magnitude," *IEEE Trans. Inf. Theory*, vol. 56, no. 3, pp. 979–983, Mar. 2010.

[31] T. Kløve, B. Bose, and N. Elarief, "Systematic single limited magnitude asymmetric error correcting codes," in *Proc. IEEE Inf. Theory Workshop Inf. Theory (ITW, Cairo)*, Jan. 2010, p. 1.

[32] E. Yaakobi, P. H. Siegel, A. Vardy, and J. K. Wolf, "On codes that correct asymmetric errors with graded magnitude distribution," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2011, pp. 1056–1060.

[33] A. V. Kuznetsov and A. J. H. Vinck, "On the general defective channel with informed encoder and capacities of some constrained memories," *IEEE Trans. Inf. Theory*, vol. 40, no. 6, pp. 1866–1871, Jun. 1994.

[34] A. A. Jiang, Y. Li, E. E. Gad, M. Langberg, and J. Bruck, "Joint rewriting and error correction in write-once memories," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 1067–1071.

[35] A. Jiang, M. Schwartz, and J. Bruck, "Correcting charge-constrained errors in the rank-modulation scheme," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2112–2120, May 2010.

[36] H. Zhou, A. Jiang, and J. Bruck, "Error-correcting schemes with dynamic thresholds in nonvolatile memories," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2011, pp. 2109–2113.

[37] H. Zhou, M. Schwartz, A. A. Jiang, and J. Bruck, "Systematic error-correcting codes for rank modulation," *IEEE Trans. Inf. Theory*, vol. 61, no. 1, pp. 17–32, Jan. 2015.

[38] L. G. Tallini and B. Bose, "On L_1-distance error control codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2011, pp. 1061–1065.

[39] L. G. Tallini and B. Bose, "On symmetric L_1 distance error control codes and elementary symmetric functions," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2012, pp. 741–745.

[40] G. Kabatiansky and S. Kruglik, "On codes correcting constant number of errors in L_1 metric," in *Proc. Inf. Technol. Syst. Conf. (ITaS)*, 2015.

[41] V. Davydov, "New class of codes, correcting errors in the lee metric," in *Proc. 16th Int. Symp. Problems Redundancy Inf. Control Syst. (REDUNDANCY)*, Oct. 2019, pp. 117–120.

[42] A. Hareedy, S. Zheng, P. Siegel, and R. Calderbank, "Efficient constrained codes that enable page separation in modern flash memories," *IEEE Trans. Commun.*, vol. 71, no. 12, pp. 6834–6848, Dec. 2023.

[43] A. Barg and A. Mazumdar, "Codes in permutations and error correction for rank modulation," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3158–3165, Jul. 2010.

[44] F. Farnoud, V. Skachek, and O. Milenkovic, "Error-correction in flash memories via codes in the Ulam metric," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3003–3020, May 2013.

[45] I. Tamo and M. Schwartz, "Correcting limited-magnitude errors in the rank-modulation scheme," *IEEE Trans. Inf. Theory*, vol. 56, no. 6, pp. 2551–2560, Jun. 2010.

[46] I. Dumer, D. Micciancio, and M. Sudan, "Hardness of approximating the minimum distance of a linear code," *IEEE Trans. Inf. Theory*, vol. 49, no. 1, pp. 22–37, Jan. 2003.

[47] A. Vardy, "The intractability of computing the minimum distance of a code," *IEEE Trans. Inf. Theory*, vol. 43, no. 6, pp. 1757–1766, Jun. 1997.

[48] U. Kapshikar and S. Kundu, "On the hardness of the minimum distance problem of quantum codes," *IEEE Trans. Inf. Theory*, vol. 69, no. 10, pp. 6293–6302, Oct. 2023.

[49] W. Banzhaf, R. S. Olson, W. Tozier, and R. Riolo, *Genetic Programming Theory and Practice XV*. Cham, Switzerland: Springer, 2018.

Anxiao (Andrew) Jiang (Senior Member, IEEE) received the B.Sc. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999, and the M.Sc. and Ph.D. degrees in electrical engineering from California Institute of Technology, Pasadena, CA, USA, in 2000 and 2004, respectively. He is currently a Professor with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA. His research interests include coding theory, data storage, and deep learning. He was a recipient of the NSF CAREER Award in 2008 for his research on information theory for flash memories, the 2009 IEEE Communications Society Data Storage Technical Committee (DSTC) Best Paper Award in signal processing and coding for data storage, the 2020 Non-Volatile Memories Workshop (NVMW) Persistent Impact Prize in information theory and coding, and the Best Paper Award at INFORMS Conference on Quality, Statistics and Reliability (ICQSR) in 2024.