Error Correction and Detection for Analog AI Computing in Edge Systems

Anxiao (Andrew) Jiang
ajiang@cse.tamu.edu
Texas A&M University
College Station, Texas, USA

Abstract

To realize the full potential of deep neural networks (DNNs) in AI-empowered edge systems, DNNs need to be much more efficient. Analog in-memory computing can potentially improve the speed and energy efficiency of AI by multiple orders, and break the "memory wall" that is currently a major bottleneck for AI. This work explores the design of analog error-correcting codes (Analog ECCs). The codes focus on the correction of errors in vector-matrix multiplications, which are a dominant part of computation in DNNs. The codes consider small but ubiquitous noise in analog edge circuits as tolerable, and focus on the correction of large errors. It presents a linear-programming based algorithm that finds the error correction/detection capabilities of codes. It also presents a number of newly discovered codes that achieve state-of-the-art performance.

Keywords

Analog Error-Correcting Code, In-Memory Computing

ACM Reference Format:
Anxiao (Andrew) Jiang. 2024. Error Correction and Detection for Analog AI Computing in Edge Systems. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD'24).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 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 [19] [20]. Let 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 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 $\mathbf{c} \in C$ 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 [19] [20]. In the following, we introduce its application to *Analog In-Memory Computing* for deep neural networks (DNNs) in edge systems. 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 [8–10, 21]. 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 [24] in current AI systems. There has been good progress in the development of analog chips in recent years, which realize DNNs for training [26] and/or inference [7] 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 [10]), 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 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 [24], phase-change memories [8, 21], etc.) at the junctions. Let $\mathbf{A} = (a_{i,j})_{L \times k}$ be a matrix of

ICCAD'24, October 27–31, 2024, Newark, NJ

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

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 [10, 17], 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 [17]), 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 [8, 10, 12, 17, 21]. 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 [19]. Let 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 C. Therefore, significant errors in $\mathbf{c}$ can be corrected by the decoder of 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 [17] [18] [19] [20].

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 C is a function

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

that returns a set of locations of UMEs. Let $\delta, \Delta \in R^+$ 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 $\mathbf{c} \in C$ 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 [17] [18]), or by estimating the values of those UMEs via an extended decoding algorithm [19].

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 [19]. 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* [19].

In this paper, we present an algorithm based on linear programming for computing the m-height of an Analog ECC, which reduces the time complexity by a factor of $(m-1)! \cdot (n-m-1)! \cdot 2^{n-m}$ compared to the baseline algorithm. It can be used to analyze not only the error-correction – but also error-detection (as defined in [19, 20]) — capabilities of analog codes. Assisted by the algorithm, we use genetic programming to search for new codes. A number of new codes that achieve state-of-the-art performance are discovered, whose m-heights are summarized in Fig. 2. The codes can be used to correct one or more (up to 4) UMEs.

2 Existing Constructions for Analog ECCs

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 [19]. 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

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

[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 [20].

[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.

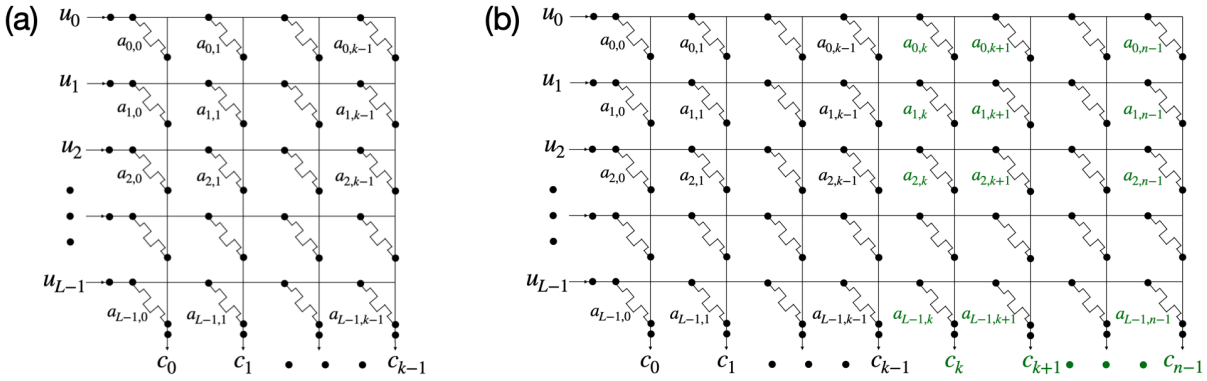

Figure 1: (a) A crossbar architecture for vector–matrix multiplication. (b) Using Analog ECC for the multiplication, where each row $(a_{i,0}, a_{i,1}, \cdots, a_{i,n-1})$ in the array and the output vector $(c_0, c_1, \cdots, c_{n-1})$ are codewords of the code.

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 C over $\mathbb{R}$ is defined as

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

The next important result was proven in [19].

Theorem 2.1. *Let C be a linear $[n, k]$ code over $\mathbb{R}$. Given $\delta, \Delta \in \mathbb{R}^+$ with $\delta < \Delta$ and a positive integer t, there exists a decoder for C that corrects t UMEs if and only if*

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

We now present the existing constructions for Analog ECCs. Let us start with the *Repetition Code* [19]. Let 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 $h_m(C) = 1$ for $m \in [n\rangle$. So by Theorem 2.1, 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* [19]. Let 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

$$G_{wk,k} = \left(\begin{array}{cccc|cc} 1 & 1 & \cdots & 1 & 0 & \cdots 0 \\ \multicolumn{4}{c|}{\mathbf{0}} & \multicolumn{2}{c}{G_{w(k-1),k-1}} \end{array} \right).$$

Its m-height is $h_m(C) = 1$ for $m \in [w\rangle$, and $h_m(C) = \infty$ for $m \geq w$. So by Theorem 2.1, 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 [19]. Although it is not for correcting any UME, it can detect a single UME by the definition of error detection in [20]. 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 C be a linear $[n, k = n - r]$ code over $\mathbb{R}$ with H as its parity-check matrix. Then its 1-height satisfies $h_1(C) \leq \lceil n/r \rceil - 1$. 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 [19], 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 [19]:

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

Let 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

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

So by Theorem 2.1, 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 [19]. 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 C satisfies

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

So by Theorem 2.1, 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 [19]. 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 C be a linear

$[n, k = n-2]$ code over $\mathbb{R}$ defined by

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

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 C satisfies

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

So by Theorem 2.1, 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, including in the joint source-channel coding (JSCC) paradigm [1, 11, 15, 22] for communications. There have also been works on ECCs for nonvolatile memories (NVMs), where codes for two or more discrete levels are studied. [3, 4, 6, 14, 16, 25]. 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 in edge systems.

3 Finding the m-Height of Analog ECC

The m-height of Analog ECC is analogous to the minimum distance of conventional ECCs (e.g., codes over finite fields), as evidenced by Theorem 2.1. It is crucial for finding the error-correction capability of a code. It is known to be NP-hard to find the minimum distance of conventional linear ECCs [5, 23]. 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 $d(C)$ of the Analog ECC C. Let $\mathrm{w}_H(\mathbf{c})$ denote the Hamming weight of a codeword $\mathbf{c}$. Then $d(C)$ equals the minimum Hamming weight of a nonzero codeword in C, namely, $d(C) = \min_{\mathbf{c}\in C-\{\mathbf{0}\}} \mathrm{w}_H(\mathbf{c})$. Based on the definition of m-height, we have

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

Let $h_n(C) \triangleq \infty$. Then $d(C)$ is the minimum index $m \in [n+1\rangle$ such that $h_m(C) = \infty$. Namely, $h_m(C) = \infty$ if and only if $m \geq d(C)$. So when the m-height values are found for all $m \in [n\rangle$, the value of $d(C)$ 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 $d(C)$, because the necessary and sufficient condition for the code to be able to correct t UMEs is $\Delta/\delta \geq 2(h_{2t}(C)+1)$, which depends on the ratio Δ/δ. The value of $d(C)$ can tell us that the code can potentially correct $\lfloor \frac{1}{2}(d(C)-1)\rfloor$ UMEs, however it cannot guarantee that any $\lfloor \frac{1}{2}(d(C)-1)\rfloor$ UMEs are correctable without knowing the ratio Δ/δ; that is, $d(C)$ 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 present an algorithm that discovers the m-height via solving $n(n-1)\binom{n-2}{m-1}2^m$ linear programs, which — compared to a baseline algorithm — reduces the complexity by a factor of $(m-1)!\cdot(n-m-1)!\cdot 2^{n-m}$.

3.1 A Baseline Algorithm for m-Height Problem

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

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

Let $C_{\pi,\mathbf{s}}$ denote the subset of codewords of C such that a codeword $\mathbf{c} = (c_0, c_1, \cdots, c_n) \in C$ is in $C_{\pi,\mathbf{s}}$ if $\mathbf{c} \neq \mathbf{0}$, $\mathrm{sgn}(c_{\pi(j)}) = s_j$ 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 $C - \{\mathbf{0}\} = \bigcup_{\pi\in\Pi,\mathbf{s}\in\mathbf{S}} C_{\pi,\mathbf{s}}$ and $h_m(\mathbf{0}) = 0$, we get

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

Lemma 3.1. *Let 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 C. Let $m \in [d(C)\rangle - \{0\}$, $\pi \in \Pi$ 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 $u_0, u_1, \cdots, u_{k-1}$:*

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

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

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

The proof for the above lemma is skipped due to space limitation.

The lemma above considers $h_m(C)$ for $m > 0$ because $h_0(C) \equiv 1$ as long as C contains a nonzero codeword. It indicates a baseline algorithm 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.

3.2 A More Efficient Algorithm for m-Height

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 $C_{a,b,X,\psi}$ denote a subset of nonzero codewords of C such that a nonzero codeword $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in C$ is in $C_{a,b,X,\psi}$ 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) $\forall\ j \in [m\rangle$, $\mathrm{sgn}(c_{\tau(j)}) = s_j$. (Note that here s_j is the j-th element of ψ.)

For any nonzero codeword $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in C$, there exists at least one tuple $(a, b, X, \psi) \in \Gamma$ such that $\mathbf{c} \in C_{a,b,X,\psi}$. (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 $\forall\ j \in [m\rangle$, $s_j = \mathrm{sgn}(c_{x_j})$. For the above tuple (a, b, X, ψ), we have $\mathbf{c} \in C_{a,b,X,\psi}$.) Therefore we have $C - \{\mathbf{0}\} = \bigcup_{(a,b,X,\psi)\in\Gamma} C_{a,b,X,\psi}$. Since $h_m(\mathbf{0}) = 0$, we get

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

Theorem 3.2. *Let 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 C where no column is $\mathbf{0}$. Let $d(C)$ be the minimum distance of C, and let $m \in \{1, 2, \cdots, \min\{d(C), n-1\}\}$. 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 $u_0, u_1, \cdots, u_{k-1}$:*

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

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

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

The proof of the theorem is skipped due to space limitation.

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

(1) $h_0(C) = 1$.
(2) For $m = 1, 2, 3\cdots$, compute $h_m(C) = \max_{a,b,X,\psi\in\Gamma} z_{a,b,X,\psi}$ 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 $h_{m^*}(C) = \infty$. We get $d(C) = m^*$.
(3) For $m = m^* + 1, m^* + 2, \cdots, n-1$, $h_m(C) = \infty$.

The above algorithm solves $n(n-1)\binom{n-2}{m-1}2^m$ linear programs of k variables to compute an $h_m(C)$. 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

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

4 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 [2]: 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.

We have used the search algorithm to construct a number of new codes. 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. A summary of the codes is shown in Fig. 2. The table lists the m-heights of different linear $[n, k]$ Analog ECCs. Here the columns labelled by "known" refer to known codes from [19], 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.

[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 $d(C) \leq n - k + 1$, so $h_m(C) = \infty$ when $m > n - k$ for any code C.

$h_m(\mathcal{C})$	$[n,k]=[5,2]$		$[n,k]=[6,2]$		$[n,k]=[6,3]$	
	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	—-	—-	—-	—-

Figure 2: The m-heights of different linear $[n,k]$ Analog ECCs.

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

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

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

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

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

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

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

Figure 3: Generator matrices of some new Analog ECCs.

Some codes in Fig. 2 have m-height equal to 1 for some m values. They are either the Repetition Code or the Cartesian Power of Repetition Code presented in [19], 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*.

As examples, the generator matrices of some new codes in Fig. 2 are shown in Fig. 3. (Due to space limitation, we skip the generator matrices of other codes here.). Each $G^m_{n,k}$ in the figure refers to the generator matrix of the new $[n,k]$ code whose m-height is shown in Fig. 2. The new codes in Fig. 2, 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.

5 Conclusions

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, an algorithm is presented for computing the m-heights of Analog ECCs, which are directly related to the codes' error-correction and error-detection capabilities. The algorithm is used to find new codes that correct one or more errors. A list of new codes with the best known performance are summarized.

Acknowledgments

This work was supported in part by the NSF Grant CCF-2416361.

References

[1] Emrah Akyol, Kumar B. Viswanatha, Kenneth Rose, and Tor A. Ramstad. 2014. On Zero Delay Source-Channel Coding. *IEEE Transactions on Information Theory* 60, 12 (2014), 7473–7489. https://doi.org/10.1109/TIT.2014.2361532

[2] Wolfgang Banzhaf, Randal S. Olson, William Tozier, and Rick Riolo. 2018. *Genetic Programming Theory and Practice XV*. Springer.

[3] Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu. 2017. Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives. *Proc. IEEE* 105, 9 (2017), 1666–1704. https://doi.org/10.1109/JPROC.2017.2713127

[4] Yuval Cassuto, Moshe Schwartz, Vasken Bohossian, and Jehoshua Bruck. 2010. Codes for Asymmetric Limited-magnitude Errors with Application to Multilevel Flash Memories. *IEEE Transactions on Information Theory* 56, 4 (2010), 1582–1595. https://doi.org/10.1109/TIT.2010.2040971

[5] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. 2003. Hardness of Approximating the Minimum Distance of a Linear Code. *IEEE Transactions on Information Theory* 49, 1 (2003), 22–37.

[6] Noha Elarief and Bella Bose. 2010. Optimal, Systematic, q-ary Codes Correcting All Asymmetric and Symmetric Errors of Limited Magnitude. *IEEE Transactions on Information Theory* 56, 3 (2010), 979–983. https://doi.org/10.1109/TIT.2009.2039065

[7] Cheng-Xin Xue *et al.* 2021. A CMOS-integrated Compute-in-memory Macro Based on Resistive Random-access Memory for AI Edge Devices. *Nature Electronics* 4, 1 (2021), 1–10.

[8] Manuel Le Gallo *et al.* 2023. A 64-core Mixed-signal In-memory Compute Chip Based on Phase-change Memory for Deep Neural Network Inference. *Nature Electronics* 6 (2023), 680–693. https://doi.org/10.1038/s41928-023-01010-1

[9] Wei Wang *et al.* 2022. A Memristive Deep Belief Neural Network Based on Silicon Synapses. *Nature Electronics* 5 (2022), 870–880.

[10] Wenbin Zhang *et al.* 2023. Edge Learning Using a Fully Integrated Neuro-inspired Memristor Chip. *Science* 381, 6663 (2023), 1205–1211. https://doi.org/DOI: 10.1126/science.ade3483

[11] Yichuan Hu, Javier Garcia-Frias, and Meritxell Lamarca. 2011. Analog Joint Source-Channel Coding Using Non-Linear Curves and MMSE Decoding. *IEEE Transactions on Communications* 59, 11 (2011), 3016–3026. https://doi.org/10.1109/TCOMM.2011.081711.090298

[12] Kunping Huang, Paul H. Siegel, and Anxiao (Andrew) Jiang. 2020. Functional Error Correction for Robust Neural Networks. *IEEE Journal on Selected Areas in Information Theory* 1, 1 (2020), 267–276. https://doi.org/10.1109/JSAIT.2020.2991430

[13] G. O. H. Katona and A. Seress. 1993. Greedy Construction of Nearly Regular Graphs. *European Journal of Combinatorics* 14, 3 (1993), 213–229.

[14] Torleiv Klove, Bella Bose, and Noha Elarief. 2010. Systematic Single Limited Magnitude Asymmetric Error Correcting Codes. In *Proceedings of IEEE Information Theory Workshop*. https://doi.org/10.1109/ITWKSPS.2010.5503196

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

[16] A. V. Kuznetsov and A. J. H. Vinck. 1994. On the General Defective Channel with Informed Encoder and Capacities of Some Constrained Memories. *IEEE Transactions on Information Theory* 40, 6 (1994), 1866–1871. https://doi.org/10.1109/18.340461

[17] Can Li, Ron M. Roth, Cat Graves, Xia Sheng, and John Paul Strachan. 2020. Analog Error Correcting Codes for Defect Tolerant Matrix Multiplication in Crossbars. In *Proceedings of IEEE International Electron Devices Meeting*. https://doi.org/10.1109/IEDM13553.2020.9371978

[18] Ron M. Roth. 2019. Fault-Tolerant Dot-Product Engines. *IEEE Transactions on Information Theory* 65, 4 (2019), 2046–2057. https://doi.org/10.1109/TIT.2018.2869794

[19] Ron M. Roth. 2020. Analog Error-Correcting Codes. *IEEE Transactions on Information Theory* 66, 7 (2020), 4075–4088.

[20] Ron M. Roth. 2022. Fault-Tolerant Neuromorphic Computing on Nanoscale Crossbar Architectures. In *Proceedings of IEEE Information Theory Workshop*.

[21] Abu Sebastian, Manuel Le Gallo, Riduan Khaddam-Aljameh, and Evangelos Eleftheriou. 2020. Memory Devices and Applications for In-memory Computing. *Nature Nanotechnology* 15 (2020). https://doi.org/10.1038/s41565-020-0655-z

[22] Claude E. Shannon. 1949. Communication in The Presence of Noise. *Proceedings of the IRE* 37, 1 (1949), 10—-21.

[23] Alexnader Vardy. 1997. The Intractability of Computing the Minimum Distance of a Code. *IEEE Transactions on Information Theory* 43, 6 (1997), 1757–1766. https://doi.org/10.1109/18.641542

[24] H.-S. Philip Wong and Sayeef Salahuddin. 2015. Memory Leads The Way to Better Computing. *Nature Nanotechnology* 10 (2015), 191–194.

[25] Eitan Yaakobi, Paul H. Siegel, Alexander Vardy, and Jack K. Wolf. 2011. On Codes That Correct Asymmetric Errors with Graded Magnitude Distribution. In *Proceedings of International Symposium on Information Theory*. 1021–1025. https://doi.org/10.1109/ISIT.2011.6033692

[26] Peng Yao, Huaqiang Wu, Bin Gao, Jianshi Tang, Qingtian Zhang, Wenqiang Zhang, J. Joshua Yang, and He Qian. 2020. Fully Hardware-implemented Memristor Convolutional Neural Network. *Nature* 577 (2020), 641–646. https://doi.org/10.1038/s41586-020-1942-4