AIMS Mathematics, 9(4): 9508–9518.
DOI:10.3934/math.2024464
Received: 31 December 2023
Revised: 09 February 2024
Accepted: 20 February 2024
Published: 07 March 2024

http://www.aimspress.com/journal/Math

Research article

Generalized Reed-Solomon codes over number fields and exact gradient coding

Irwansyah[1,*], **Intan Muchtadi-Alamsyah**[2], **Fajar Yuliawan**[2] **and Muhammad Irfan Hidayat**[2]

[1] Department of Mathematics, Faculty of Mathematics and Natural Sciences, University of Mataram, Indonesia

[2] Algebra Research Group, Faculty of Mathematics and Natural Sciences, Institut Teknologi Bandung, Indonesia

* **Correspondence:** Email: irw@unram.ac.id.

Abstract: This paper describes generalized Reed-Solomon (GRS) codes over number fields that are invariant under certain permutations. We call these codes generalized quasi-cyclic (GQC) GRS codes. Moreover, we describe an application of GQC GRS codes over number fields to exact gradient coding.

Keywords: generalized Reed-Solomon codes; generalized quasi-cyclic codes; number fields; Galois group; exact gradient coding
Mathematics Subject Classification: 11T71, 68P30

1. Introduction

Data-intensive machine learning has become widely used, and as the size of training data increases, distributed methods are becoming increasingly popular. However, the performance of distributed methods is mainly determined by stragglers, i.e., nodes that are slow to respond or are unavailable.

Raviv et al. [11] used coding theory and graph theory to reduce stragglers in distributed synchronous gradient descent. A coding theory framework for straggler mitigation, called gradient coding, was first introduced by Tandon et al. [14]. Gradient coding consists of a system with one master and n worker nodes, where the data are partitioned into k parts, and one or more parts are assigned to each worker. In turn, each worker computes the partial gradients on each given partition, combines the results linearly according to a predefined vector of coefficients, and sends this linear combination back to the primary node. By choosing the coefficients at each node appropriately, it can be guaranteed that the primary node can reconstruct the full gradient even if a machine fails to do its job.

The importance of straggler mitigation is demonstrated in [8, 16]. Specifically, it was shown by Tandon et al. [14] that stragglers run up to 5 times slower than the performance of typical workers (8

times in [16]). In [11], for gradient calculations, a cyclic maximum distance separable (MDS) code is used to obtain a better deterministic construction scheme than existing solutions, both in the range of parameters that can be applied and in the complexity of the algorithms involved.

One well-known family of MDS codes is generalized Reed-Solomon (GRS) codes. GRS codes have interesting mathematical structures and many real-world applications, such as mass storage systems, cloud storage systems, and public-key cryptosystems. On the other hand, although more complex than cyclic codes, quasi-cyclic codes satisfy the condition of the Gilbert-Varshamov lower bound at minimum distances, as shown in [6]. Quasi-cyclic codes are also equivalent to linear codes with circulant block generator matrices. This type of matrix has circular blocks of the same size, such as m, which denotes the co-indexes of the associated quasi-cyclic code. From this point of view, one way to generalize quasi-cyclic codes is to let the generator matrix have circular blocks of different sizes. This code is called a generalized quasi-cyclic code with shared indices $(m_1, m_2, \ldots, m_k)$, where $m_1, m_2, \ldots, m_k$ represents the size of the circular block in the generator matrix.

In [10], a generalized quasi-cyclic code without block length limitations is studied. By relaxing the conditions on block length, several new optimal codes with small lengths could be found. In addition, the code decomposition and dimension formulas given by [3, 12, 13] have been generalized.

In this paper, we describe the construction of generalized quasi-cyclic GRS codes over totally real number fields, as well as their application in exact gradient coding. The construction method is derived by integrating known results from the inverse Galois problem for totally real number fields. Furthermore, methods in [2,4,11,14] will be adapted to generalized quasi-cyclic GRS codes to mitigate stragglers.

2. Generalized Reed-Solomon codes

Let $\mathbb{F}$ be a Galois extension of $\mathbb{Q}$ and choose non-zero elements $v_1, \ldots, v_n$ in $\mathbb{F}$ and distinct elements $a_1, \ldots, a_n$ in $\mathbb{F}$. Also, let $\mathbf{v} = (v_1, \ldots, v_n)$ and $\mathbf{a} = (a_1, \ldots, a_n)$. For $1 \leq k \leq n$, define the GRS codes as follows:

$$GRS_{n,k}(\mathbf{a}, \mathbf{v}) = \{(v_1 f(a_1), \ldots, v_n f(a_n)) \mid f(x) \in \mathbb{F}[x]_k\},$$

where $\mathbb{F}[x]_k$ is the set of all polynomials over $\mathbb{F}$ with degree less than k. The canonical generator of $GRS_{n,k}(\mathbf{a}, \mathbf{v})$ is given by the following matrix:

$$\mathbf{G} = \begin{pmatrix} v_1 & v_2 & \cdots & v_j & \cdots & v_n \\ v_1 a_1 & v_2 a_2 & \cdots & v_j a_j & \cdots & v_n a_n \\ v_1 a_1^2 & v_2 a_2^2 & \cdots & v_j a_j^2 & \cdots & v_n a_n^2 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v_1 a_1^i & v_2 a_2^i & \cdots & v_j a_j^i & \cdots & v_n a_n^i \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v_1 a_1^{k-1} & v_2 a_2^{k-1} & \cdots & v_j a_j^{k-1} & \cdots & v_n a_n^{k-1} \end{pmatrix} \tag{2.1}$$

Theorem 2.1. *[7] Let $\mathbf{v} \in \mathbb{F}^n$ be a tuple of non-zero elements in $\mathbb{F}$ and $\mathbf{a} \in \mathbb{F}^n$ be a tuple of pairwise distinct elements in $\mathbb{F}$; then,*

a) The $GRS_{n,k}(\mathbf{a}, \mathbf{v})$ is a $[n, k, n-k+1]$ code, i.e., GRS codes are MDS codes.

b) The dual code of $GRS_{n,k}(\mathbf{a}, \mathbf{v})$ is as follows:

$$GRS_{n,k}(\mathbf{a}, \mathbf{v})^{\perp} = GRS_{n,n-k}(\mathbf{a}, \mathbf{u}),$$

where $\mathbf{u} = (u_1, \ldots, u_n)$ with

$$u_i^{-1} = v_i \prod_{j \neq i} \left(a_i - a_j\right).$$

Proof. (a) See the proof of [7, Theorem 6.3.3]. (b) See the proof of [7, Theorem 6.5.1].

Let $\overline{\mathbb{F}} = \mathbb{F} \cup \{\infty\}$ and $\mathbf{a}$ be an n-tuple of mutually distinct elements of $\overline{\mathbb{F}}$, and let $\mathbf{c}$ be an n-tuple of non-zero elements of $\mathbb{F}$. Also, define

$$[a_i, a_j] = a_i - a_j, \quad [\infty, a_j] = 1 \quad [a_i, \infty] = -1 \quad \text{for all } a_i, a_j \in \mathbb{F}.$$

Definition 2.2. *([9]) Let $B(\mathbf{a}, \mathbf{c})$ be the $k \times (n-k)$ matrix with the following entries:*

$$\frac{c_{j+k}}{c_i[a_{j+k}, a_i]}, \text{for } 1 \leq i \leq k, 1 \leq j \leq n-k.$$

The generalized Cauchy code $C_k(\mathbf{a}, \mathbf{c})$ is an $[n, k, n-k+1]$ code defined by the generator matrix $(I_k \mid B(\mathbf{a}, \mathbf{c}))$.

The following proposition shows that the GRS codes are also generalized Cauchy codes.

Proposition 2.3. *[9, Proposition C.2] Let $\mathbf{a}$ be an n-tuple of mutually distinct elements of $\overline{\mathbb{F}}$, and let $\mathbf{c}$ be an n-tuple of non-zero elements of $\mathbb{F}$. Also, let*

$$c_i = \begin{cases} b_i \prod_{t=1, t\neq i}^{k} [a_i, a_t], & \text{if } 1 \leq i \leq k; \\ b_i \prod_{t=1}^{k} [a_i, a_t], & \text{if } k+1 \leq i \leq n. \end{cases}$$

Then, $GRS_{n,k}(\mathbf{a}, \mathbf{b}) = C_k(\mathbf{a}, \mathbf{c})$.

Let $Gal(\mathbb{F}/\mathbb{Q})$ be the Galois group of $\mathbb{F}$ over $\mathbb{Q}$ and $P\Gamma L(2, \mathbb{F})$ denote the group of semilinear fractional transformations given by

$$\begin{array}{rccc} f: & \overline{\mathbb{F}} & \longrightarrow & \overline{\mathbb{F}} \\ & x & \longmapsto & \dfrac{a\gamma(x) + b}{c\gamma(x) + d}, \end{array}$$

where $ad - bc \neq 0$ and $\gamma \in Gal(\mathbb{F}/\mathbb{Q})$. Let S_n be the symmetric group on a set of n elements and $Per(C) = \{\xi \in S_n \mid \xi(C) = C\}$, where n is the length of the code C. The set $Per(C)$ is called the permutation group of the code C. We have the following theorem that is related to the permutation group of a Cauchy code.

Theorem 2.4. *[1, Corollary 2] Let $C = C_k(\mathbf{a}, \mathbf{y})$ be a Cauchy code over $\mathbb{F}$, where $2 \leq k \leq n-2$ and $\mathbf{a} = (a_1, \ldots, a_n)$. Also, let $L = \{a_1, \ldots, a_n\}$. Then, the map*

$$\begin{array}{rccc} \omega: & \{f \in P\Gamma L(2, \mathbb{F}) \mid f(L) = L\} & \longrightarrow & Per(C) \\ & f & \longmapsto & \sigma, \end{array}$$

where $a_{\sigma(i)} = f(a_i)$ for $i = 1, \ldots, n$ is a surjective group homomorphism.

3. Galois group of a number field

A number field $\mathbb{F}$ is a finite Galois extension of the rational field $\mathbb{Q}$. In this section, we describe a way to construct a number field $\mathbb{F}$ with $Gal(\mathbb{F}/\mathbb{Q}) \cong \langle\sigma\rangle$ for $\sigma \in S_n$, where $\langle\sigma\rangle$ is a cyclic subgroup generated by σ.

Let $\sigma = \sigma_1\sigma_2\cdots\sigma_t$ be a permutation in S_n, where $\sigma_1, \sigma_2, \ldots, \sigma_t$ are disjoint cycles. Also, let $\langle\sigma\rangle$ be the cyclic group generated by σ. Let $l(\sigma_j)$ be the length of the cycle σ_j, and define a set $\mathcal{P} = \left\{p : p \text{ prime and } \exists j \in \{1, \ldots, t\} \ni p|l(\sigma_j)\right\}$. Since $\mathcal{P}$ is finite, assume that $p_1 < p_2 < \cdots < p_{|\mathcal{P}|}$ are all elements in $\mathcal{P}$. For any j, we have

$$l(\sigma_j) = \prod_{i=1}^{|\mathcal{P}|} p_i^{\alpha_{ij}}, \tag{3.1}$$

where $\alpha_{ij} \in \mathbb{Z}_{\geq 0}$. Based on Eq (3.1), we have

$$ord(\sigma) = |\langle\sigma\rangle| = \prod_{i=1}^{|\mathcal{P}|} p_i^{\max_j\{\alpha_{ij}\}}, \tag{3.2}$$

where $ord(\sigma)$ is the order of the permutation σ. Since $\langle\sigma\rangle$ contains the element of order $p_i^{\max_j\{\alpha_{ij}\}}$ for all $i = 1, \ldots, |\mathcal{P}|$, by the structure theorem for finite Abelian groups, we have

$$\langle\sigma\rangle \cong \prod_{i=1}^{|\mathcal{P}|} \frac{\mathbb{Z}}{p_i^{\max_j\{\alpha_{ij}\}}\mathbb{Z}} \cong \frac{\mathbb{Z}}{\prod_{i=1}^{|\mathcal{P}|} p_i^{\max_j\{\alpha_{ij}\}}\mathbb{Z}}. \tag{3.3}$$

Let ζ_p be the primitive p-th root of unity and $\mathbb{Q}(\zeta_p)$ be the corresponding cyclotomic extension of $\mathbb{Q}$. The following theorem shows a Galois extension of $\mathbb{Q}$, where its Galois group is isomorphic to $\langle\sigma\rangle$. The proof of the theorem is similar to the proof of [5, Theorem 3.1.11]. We write the proof here to give a sense of how to construct the related Galois extension.

Theorem 3.1. *There exists a totally real Galois extension K of $\mathbb{Q}$ such that $Gal(K/\mathbb{Q}) \cong \langle\sigma\rangle$.*

Proof. By Eq (3.3), we have

$$\langle\sigma\rangle \cong \prod_{i=1}^{|\mathcal{P}|} \frac{\mathbb{Z}}{p_i^{\max_j\{\alpha_{ij}\}}\mathbb{Z}} \cong \frac{\mathbb{Z}}{\prod_{i=1}^{|\mathcal{P}|} p_i^{\max_j\{\alpha_{ij}\}}\mathbb{Z}}.$$

Now, choose a prime p such that

$$p \equiv 1 \mod 2\prod_{i=1}^{|\mathcal{P}|} p_i^{\max_j\{\alpha_{ij}\}}.$$

Let ζ_p be the p-th root of unity. By [5, Theorem C.0.3], $\mathbb{Q}(\zeta_p)$ is a Galois extension of $\mathbb{Q}$, with its corresponding Galois group being isomorphic to $G = (\mathbb{Z}/p\mathbb{Z})^\times$, where $(\mathbb{Z}/p\mathbb{Z})^\times$ is the multiplicative group of $\mathbb{Z}/p\mathbb{Z} - \left\{\overline{0}\right\}$. Since p is a prime number, G is a cyclic group. Moreover, we can find a unique subgroup H of G such that

$$|H| = \frac{p-1}{\prod_{i=1}^{|\mathcal{P}|} p_i^{\max_j\{\alpha_{ij}\}}}.$$

Let $\mathbb{Q}(\zeta_p)^H$ be a subset of $\mathbb{Q}(\zeta_p)$ which is invariant under the action of H. By the fundamental theorem of Galois theory ([15, Theorem 25]), $\mathbb{Q}(\zeta_p)^H$ is also a Galois extension of $\mathbb{Q}$, with the corresponding Galois group isomorphic to G/H. Moreover, $|G/H| = \prod_{i=1}^{|\mathcal{P}|} p_i^{\max_j\{\alpha_{ij}\}}$, and, as a consequence,

$$G/H \cong \frac{\mathbb{Z}}{\prod_{i=1}^{|\mathcal{P}|} p_i^{\max_j\{\alpha_{ij}\}}\mathbb{Z}} \cong \langle\sigma\rangle.$$

Also, by using a similar argument as in the proof of [5, Theorem 3.1.11], we have that $\mathbb{Q}(\zeta_p)^H$ is a totally real Galois extension of $\mathbb{Q}$.

The following algorithm provides a way to construct $\mathbb{Q}(\zeta_p)^H$ in the proof of Theorem 3.1. The algorithm is based on Theorem 3.1 and [5, Proposition 3.3.2].

Algorithm 3.2. *Suppose that $\sigma \in S_n$ and $G = Gal(\mathbb{Q}(\zeta_p)/\mathbb{Q}) \cong (\mathbb{Z}/p\mathbb{Z})^\times$, where p is a prime number such that $p \equiv 1 \mod 2 \cdot ord(\sigma)$.*

1) Choose $H \subseteq G$, where H is the subgroup of G with order $\dfrac{p-1}{ord(\sigma)}$.
2) Calculate
$$\alpha = \sum_{\lambda\in H} \lambda(\zeta_p).$$
3) Find minimal polynomial $m_\alpha(x)$ of α over $\mathbb{Q}$.
4) Construct the splitting field $\mathbb{F}$ of $m_\alpha(x)$ by using Algorithm A.1.
5) Then, $\mathbb{F} = \mathbb{Q}(\zeta_p)^H$.

4. GRS GQC codes over number fields

In this section, we describe a way to construct an invariant GRS code under a given permutation in S_n. We call this GRS code the GRS generalized quasi-cyclic (GQC) code. Let $\sigma = \sigma_1, \sigma_2, \cdots, \sigma_t$ be a permutation in S_n, where $\sigma_1, \sigma_2, \ldots, \sigma_t$ are disjoint cycles. Also, let $G = \langle\sigma\rangle$ be a cyclic group generated by σ.

Theorem 4.1. *If σ is a permutation in S_n, then there exists a GQC $GRS_{n,k}(\overline{\alpha}, \mathbf{b})$ over $\mathbb{F}$, with its corresponding permutation being σ for some totally real number field $\mathbb{F}$.*

Proof. We can find the number field $\mathbb{F}$ and its corresponding minimal polynomial $m_\alpha(x)$ with $Gal(\mathbb{F}/\mathbb{Q}) \cong \langle\sigma\rangle$ by using Algorithm 3.2. Since $Gal(\mathbb{F}/\mathbb{Q}) \cong \langle\sigma\rangle$, there exists $\gamma \in Gal(\mathbb{F}/\mathbb{Q})$ to be associated with $\sigma \in \langle\sigma\rangle$. Let $L = \{\alpha_1, \ldots, \alpha_n\}$ be the roots of $m_\alpha(x)$ and some additional elements from linear combinations of the roots. We can see that γ is a permutation on L, i.e., $\gamma(L) = L$. Note that the orbit of L under H can be used to rearrange the elements of L such that

$$\gamma(\alpha_i) = \alpha_{\sigma(i)}, \tag{4.1}$$

for all $i = 1, 2, \ldots, n$. Let $\overline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ be an n-tuple of non-zero elements in $\mathbb{F}$. Define a Cauchy code $C_k(\overline{\alpha}, \mathbf{c})$, where $\mathbf{c} = (c_1, c_2, \ldots, c_n)$, with

$$c_i = \begin{cases} b_i \prod_{t=1, t\neq i}^{k} [\alpha_i, \alpha_t], & \text{if } 1 \leq i \leq k; \\ b_i \prod_{t=1}^{k} [\alpha_i, \alpha_t], & \text{if } k+1 \leq i \leq n. \end{cases} \tag{4.2}$$

Then, by Proposition 2.3, $C_k(\overline{\alpha}, \mathbf{c})$ is a $GRS_{n,k}(\overline{\alpha}, \mathbf{b})$ code. Moreover, according to Theorem 2.4 and Eq (4.1), $\omega(\gamma) = \sigma$ is an element in $Per\left(C_k(\overline{\alpha}, \mathbf{c})\right)$.

Consider the following example.

Example 4.2. *Let $\sigma = (1, 2, 3, 4)(5, 6)$ in S_6. We would like to construct a GRS code of length 6 over a totally real number field that is invariant under the action of σ. We can see that $ord(\sigma) = 4$ and $\langle\sigma\rangle = \mathbb{Z}/4\mathbb{Z}$. Choose $p = 17$ so that $p \equiv 1 \mod 2\times 4$. The corresponding subgroup H of $Gal(\mathbb{Q}(\zeta_{17})/\mathbb{Q})$ will have the order equal to 4. Since the unique subgroup of $(\mathbb{Z}/17\mathbb{Z})^{\times}$ with order 4 is $\{1, 4, 13, 16\}$, we have*

$$H = \{\lambda_k | k = 1, 4, 13, 16\},$$

where $\lambda_k : \zeta_{17} \mapsto \zeta_{17}^k$. Then, we have

$$\alpha = \sum_{\lambda \in H} \lambda(\zeta_{17}) = \zeta_{17} + \zeta_{17}^4 + \zeta_{17}^{13} + \zeta_{17}^{16}.$$

From [5, Example 3.3.3], the minimal polynomial of α is as follows:

$$m_\alpha(x) = x^4 + x^3 - 6x^2 - x + 1.$$

The roots of $m_\alpha(x)$ given by

$$r_1 = \frac{1}{4}\left(-1 - \sqrt{17} - \sqrt{34 + \sqrt{17}}\right), \quad r_2 = \frac{1}{4}\left(-1 - \sqrt{17} + \sqrt{34 + \sqrt{17}}\right),$$

$$r_3 = \frac{1}{4}\left(-1 + \sqrt{17} - \sqrt{34 - \sqrt{17}}\right)., \quad r_4 = \frac{1}{4}\left(-1 + \sqrt{17} + \sqrt{34 - \sqrt{17}}\right).$$

Let γ be a map such that

$$r_1 \mapsto r_2, \quad r_2 \mapsto r_3, \quad r_3 \mapsto r_4, \quad r_4 \mapsto r_1.$$

We can see that $\langle\gamma\rangle = Gal(\mathbb{Q}(\zeta_{17})^H/\mathbb{Q}) \cong \mathbb{Z}/4\mathbb{Z}$.

Choose $L = \{\alpha_1, \ldots, \alpha_6\}$, where $\alpha_i = r_i$ for $i = 1, 2, 3, 4$, $\alpha_5 = r_1 + r_3$, and $\alpha_6 = r_2 + r_4$. We can check that

$$\gamma(\alpha_i) = \alpha_{\sigma(i)},$$

for all $i = 1, \ldots, 6$. Take $\overline{\alpha} = (\alpha_1, \ldots, \alpha_6)$, any n-tuple of non-zero elements $\mathbf{b}$ (from the set of linear combinations of roots of $m_\alpha(x)$), and $\mathbf{c} = (c_1, \ldots, c_6)$, where c_i is as in Eq 4.2. We have that $C_k(\overline{\alpha}, \mathbf{c})$ is a GQC GRS code with corresponding permutation σ.

5. Exact gradient coding using GRS codes

In Section 4, we described the construction of GRS code, which is invariant under the action of a given permutation in S_n. Moreover, the alphabet for the corresponding codes is a totally real number field, not a complex number field. This feature can be useful for bandwidth reduction in exact gradient coding schemes.

Algorithm 1 describes the process of gradient coding. The algorithm is a slight modification of [11, Algorithm 1].

Algorithm 1 Gradient coding

Input:

Data $\mathcal{S} = \{z_i = (x_i, y_i)\}_{i=1}^m$, number of iterations $t > 0$, learning rate $\{\eta\}_{r=1}^t$, straggler tolerance parameter $\{s_r\}_{r=1}^t$, a matrix $\mathbf{B} \in \mathbb{C}^{n\times n}$, a function $\Lambda : \mathcal{P}(n) \to \mathbb{C}^n$, a vector of non-zero elements $\overline{\beta} = (\beta_1, \ldots, \beta_n) \in \mathbb{C}^n$

Initialize:

$\mathbf{w}^{(1)} \leftarrow (0, 0, \ldots, 0)$

Partition $\mathcal{S} = \bigcup_{i=1}^n \mathcal{S}_i$ and send $\{\mathcal{S}_j | j \in supp(\mathbf{b}_i)\}$ to W_i for every $i \in [n]$

for $r = 1$ to t **do**

M broadcasts $\mathbf{w}^{(r)}$ to all nodes

Each W_j sends $\sum_{i\in supp(\mathbf{b}_j)} b_{j,i} \frac{\nabla L_{\mathcal{S}_i}(\mathbf{w}^{(r)})}{\beta_i}$ to M

M waits until at least $n - s_r$ nodes have responded

M computes $\mathbf{v}_r = \Lambda(\mathcal{K}_r) \cdot \mathbf{C}$, where the i-th row of $\mathbf{C}$ is $\frac{1}{n}$ times the response from W_i if it has responded, and 0 otherwise; also, $\mathcal{K}_r$ is the set of non-stragglers in the current iteration r

M updates $\mathbf{w}^{(r+1)} \leftarrow \mathbf{w}^{(r)} - \eta_r \mathbf{v}_r$

end for

return $\frac{1}{t} \sum_{r=1}^t \mathbf{w}^{(r+1)}$

Algorithm 1 works in the following way. In order to execute the gradient descent process, the master node M distributes a particular partition of the training set $\mathcal{S}$ to all worker nodes W_j, where $j = 1, \ldots, n$. In the r-th iteration of the gradient descent process, the master M broadcasts the parameter $\mathbf{w}^{(r)}$ to all worker nodes. Using the received parameter $\mathbf{w}^{(r)}$, the worker node W_j calculates the partial gradient $\nabla L_{\mathcal{S}_i}(\mathbf{w}^{(r)})$ and sends its linear combination $\sum_{i\in supp(\mathbf{b}_j)} b_{j,i} \frac{\nabla L_{\mathcal{S}_i}(\mathbf{w}^{(r)})}{\beta_i}$ to M. The linear combination is chosen from the entries $b_{j,i}$ of a particular matrix $\mathbf{B}$. In this work, $\mathbf{B}$ is constructed by using GRS codes which are invariant under the action of a particular permutation. After M has received the linear combinations of partial gradients from some number of worker nodes, M updates the parameter $\mathbf{w}$ by using the decoding vector $\Lambda(\mathcal{K}_r)$, $\mathbf{w}^{(r)}$, and some other additional vectors (mentioned in the algorithm). Note that we will see later that the decoding vector $\Lambda(\mathcal{K}_r)$ can be computed by using Algorithm 2 [11, Algorithm 2].

Definition 5.1. *A matrix $\mathbf{B} \in \mathbb{C}^{n\times n}$ and a function $\Lambda : \mathcal{P}(n) \to \mathbb{C}^n$ satisfy the exact computation (EC) condition with respect to $\overline{\beta} \in \mathbb{C}^n$, where $\overline{\beta}$ is an n-tuple of non-zero elements in $\mathbb{C}^n$ if, for all $\mathcal{K} \subseteq [n]$ such that $|\mathcal{K}| \geq \max_{r\in[t]} s_r$, we have that $\Lambda(\mathcal{K}) \cdot \mathbf{B} = \overline{\beta}$.*

Note that Definition 5.1 is a slight modification of [11, Definition 2]. Let $\overline{\beta} = (\beta_1, \ldots, \beta_n)$ be an

n-tuple of non-zero elements of $\mathbb{C}^n$ and

$$\mathbf{N}_{\overline{\beta}}(\mathbf{w}) = \frac{1}{n}\begin{pmatrix} \frac{\nabla L_{\mathcal{S}_1}(\mathbf{w})}{\beta_1} \\ \frac{\nabla L_{\mathcal{S}_2}(\mathbf{w})}{\beta_2} \\ \vdots \\ \frac{\nabla L_{\mathcal{S}_n}(\mathbf{w})}{\beta_n} \end{pmatrix}.$$

Lemma 5.2. *If Λ and $\mathbf{B}$ satisfy the EC condition with respect to $\overline{\beta}$, then, for all $r \in [t]$, we have that $\mathbf{v}_r = \nabla L_{\mathcal{S}}(\mathbf{w}^{(r)})$.*

Proof. Given $r \in [t]$, let $\mathbf{B}'$ be the matrix whose i-th row $\mathbf{b}'_i$ equals to $\mathbf{b}_i$ if $i \in \mathcal{K}_r$, and $\mathbf{0}$ otherwise. The matrix $\mathbf{C}$ in Algorithm 1 can be written as $\mathbf{C} = \mathbf{B}' \cdot \mathbf{N}_{\overline{\beta}}(\mathbf{w}^{(r)})$. Since $supp\left(\Lambda\left(\mathcal{K}_r\right)\right) \subseteq \mathcal{K}_r$, we have that $\Lambda\left(\mathcal{K}_r\right) \cdot \mathbf{B}' = \Lambda\left(\mathcal{K}_r\right) \cdot \mathbf{B}$. Therefore, we have

$$\begin{aligned}
\mathbf{v}_r &= \Lambda\left(\mathcal{K}_r\right) \cdot \mathbf{C} \\
&= \Lambda\left(\mathcal{K}_r\right) \cdot \mathbf{B} \cdot \mathbf{N}_{\beta}(\mathbf{w}^{(r)}) \\
&= \beta \cdot \mathbf{N}_{\beta}(\mathbf{w}^{(r)}) \\
&= \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \nabla L_{\mathcal{S}_i}\left(\mathbf{w}^{(r)}\right) \\
&= \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \tfrac{1}{m/n} \sum_{z \in \mathcal{S}_i} \nabla l\left(\mathbf{w}^{(r)}, z\right) \\
&= \tfrac{1}{m} \textstyle\sum_{z \in \mathcal{S}} \nabla l\left(\mathbf{w}^{(r)}, z\right) \\
&= \nabla L_{\mathcal{S}}\left(\mathbf{w}^{(r)}\right).
\end{aligned}$$

For a given n and s, let $C = GRS_{n,n-s}(\overline{\alpha}, \overline{\beta})$ GQC code over a number field $\mathbf{F}$ with corresponding permutation π of order n. Clearly, the vector $\overline{\beta}$ is in C. Moreover, by [11, Lemma 8], there exists a codeword $\mathbf{c}_1$ in C whose support is $\{1, 2, \ldots, s+1\}$. Let $\mathbf{c}_i = \pi^{i-1}(\mathbf{c}_1)$ for $i = 2, \ldots, n$ and $\mathbf{B} = \left(\mathbf{c}_1^T, \mathbf{c}_2^T, \ldots, \mathbf{c}_n^T\right)$.

Theorem 5.3. *The matrix $\mathbf{B}$ satisfies the following properties:*

a) Each row of $\mathbf{B}$ is a codeword in $\sigma(C)$, where σ is a permutation such that

$$\sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & \cdots & i & \cdots & n \\ n & \pi^{n-1}(n) & \pi^{n-2}(n) & \cdots & \pi^{n-(i-1)}(n) & \cdots & \pi(n) \end{pmatrix}. \tag{5.1}$$

b) $w_H(\mathbf{b}) = s + 1$ for each row $\mathbf{b}$ in $\mathbf{B}$.
c) The column span of $\mathbf{B}$ is the code C.
d) Every set of $n - s$ rows of $\mathbf{B}$ are linearly independent over $\mathbb{F}$.

Proof. (a) Let $\mathbf{c}_1 = (c_1, \ldots, c_n)$. Notice that the i-th row of $\mathbf{B}$ is as follows:

$$\left(c_i, c_{\pi^{n-1}(i)}, c_{\pi^{n-2}(i)}, \ldots, c_{\pi(i)}\right).$$

Since $ord(\pi) = n$, the i-th row of $\mathbf{B}$ is a permutation of $\mathbf{c}_1$ for all $i = 1, \ldots, n$. Moreover, by considering the last row of $\mathbf{B}$, we can see that all rows of $\mathbf{B}$ constitute a codeword in $\sigma(C)$, where σ is the permutation as in Eq (5.1).

(b) By part (a), we have that the Hamming weight of every row of $\mathbf{B}$ is $s+1$.

(c) Let $\sigma = (1, 2, \ldots, n)$ be a cyclic permutation and G_1 be a cyclic group generated by σ. Also, let G_2 be a cyclic group generated by π. Define $\overline{S}_1 = span(G_1\mathbf{c}_1)$ and $\overline{S}_2 = span(G_2\mathbf{c}_1)$, where $G\mathbf{c}_1 = \{\lambda(\mathbf{c}_1)|\lambda \in G\}$. Since $ord(\sigma) = ord(\pi) = n$, we have that $G_1 \cong G_2$ by the following group isomorphism:

$$\begin{array}{rccc} \tau: & G_1 & \to & G_2 \\ & \sigma^i & \mapsto & \pi^i. \end{array}$$

Define the following map:

$$\begin{array}{rccc} \overline{\tau}: & \overline{S}_1 & \to & \overline{S}_2 \\ & \sum_{i=1}^{n} \alpha_i \sigma^i(\mathbf{c}_1) & \mapsto & \sum_{i=1}^{n} \alpha_i \pi^i(\mathbf{c}_1). \end{array}$$

The map $\overline{\tau}$ is a linear map. Since it is induced by τ, $\overline{\tau}$ is a bijective map. So, $\overline{S}_1 \cong \overline{S}_2$. By [11, Lemma 12 B3], $\overline{S}_1 = C$. Since $\overline{S}_2 \subseteq C$ and $dim\ \overline{S}_2 = n - s$, we have that $\overline{S}_2 = C$.

(d) Similar to [11, Lemma 12 B4].

Let $\mathbf{G}$ be the canonical generator for the $C = GRS_{n,n-s}(\overline{\alpha}, \overline{\beta})$ GQC code, as in Eq (2.1). By Theorem 2.1(b), the canonical generator for the dual code $C^{\perp}$ is $\mathbf{G}^{\perp} = \mathbf{G} \cdot \mathbf{D}$, where $\mathbf{D} = diag(u_1, \ldots, u_n)$, with

$$u_i = \frac{1}{\beta_i^2 \prod_{j \neq i} \left(\alpha_i - \alpha_j\right)}$$

for all $i = 1, \ldots, n$. Using this setting, Algorithm 2 [11, Algorithm 2] can be used to compute the decoding vector $a(\mathcal{K})$.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This research was funded by Hibah PPMI KK Aljabar Institut Teknologi Bandung 2023.

Conflict of interest

The authors declare no conflict of interest.

References

1. A. Dür, The automorphism groups of Reed-Solomon codes, *J. Comb. Theory A*, **44** (1987), 69–82. https://doi.org/10.1016/0097-3165(87)90060-4
2. S. Dutta, V. Cadambe, P. Grover, Short-dot: Computing large linear transforms distributedly using coded short dot products, *IEEE T. Inform. Theory*, **65** (2019), 6171–6193. https://doi.org/10.1109/TIT.2019.2927558

3. C. Güneri, F. Özbudak, B. Özkaya, E. Sacikara, Z. Sepasdar, P. Solé, Structure and performance of generalized quasi-cyclic codes, *Finite Fields Th. App.*, **47** (2017), 183–202. https://doi.org/10.1016/j.ffa.2017.06.005
4. W. Halbawi, Error-correcting codes for networks, storage, and computation, PhD Thesis, California Institute of Technology, 2017. https://doi.org/10.7907/Z9J67F08
5. S. Kalyanswamy, Inverse Galois problem for totally real number fields, PhD Thesis, Cornell University, 2012.
6. T. Kasami, A Gilberd-Varshamov bound for quasi cyclic codes of rate 1/2, *IEEE T. Inform. Theory*, **20** (1974), 679. https://doi.org/10.1109/TIT.1974.1055262
7. S. Loepp, W. K. Wooters, *Protecting information: from classical error correction to quantum cryptography*, Cambridge: Cambridge University Press, 2006. https://doi.org/10.1017/CBO9780511813719
8. M. Li, D. G. Andersen, A. Smola, K. Yu, Communication efficient distributed machine learning with the parameter server, *Proceedings of the 27th International Conference on Neural Information Processing Systems*, **1** (2014), 19–27.
9. I. Márquez-Corbella, R. Pellikaan, A characterization of MDS codes that have an error-correcting pair, *Finite Fields Th. App.*, **40**, (2016), 224–245. https://doi.org/10.1016/j.ffa.2016.04.004
10. I. Muchtadi-Alamsyah, Irwansyah, A. Barra, Generalized quasi-cyclic codes with arbitrary block lengths, *Bull. Malays. Math. Sci. Soc.*, **45** (2022), 1383–1407. https://doi.org/10.1007/s40840-022-01251-x
11. N. Raviv, I. Tamo, R. Tandon, A. G. Dimakis, Gradient coding from cyclic MDS codes and expander graph, *IEEE T. Inform. Theory*, **66** (2020), 7475–7489. https://doi.org/10.1109/TIT.2020.3029396
12. G. E. Séguin, The algebraic structure of codes invariant under a permutation, In: *Information theory and applications II*, Heidelberg: Springer, 1995, 1–18. https://doi.org/10.1007/BFb0025131
13. I. Siap, N. Kulhan, The structure of generalized quasi-cyclic codes, *Appl. Math. E-Notes*, **5** (2005), 24–30.
14. R. Tandon, Q. Lei, A. Dimakis, N. Karampatziakis, Gradient coding: Avoiding stragglers in distributed learning, *Proceedings of the 34th International Conference on Machine Learning*, **70** (2017), 3368–3376.
15. H. G. J. Tiesinga, The inverse Galois problem, PhD Thesis, University of Groningen, 2016. Available from:
16. N. J. Yadwarkar, B. Hariharan, J. E. Gonzales, R. Katz, Multitask learning for straggler avoiding predictive job scheduling, *J. Mach. Learn. Res.*, **17** (2016), 3692–728.

A. Splitting field of a polynomial

The following algorithm provides a way to construct the unique splitting field of a given polynomial $f(x)$ in $\mathbb{Q}[x]$.

Algorithm A.1. *Given a polynomial $f(x)$ in $\mathbb{Q}[x]$, we will construct the splitting field L of $f(x)$ based on the construction of a chain of number fields:*

$$K_0 = \mathbb{Q} \subset K_1 \subset K_2 \subset \cdots \subset K_{s-1} \subset K_s = L$$

such that K_i is an extension of K_{i-1} containing a new root of $f(x)$.

1) Factorize $f(x)$ over K_i into irreducible factors $f_1(x)f_2(x)\cdots f_t(x)$.
2) Choose any non linear irreducible factor $g(x) = f_j(x)$ for some $j \in \{1, \ldots, t\}$.
3) Construct the field extension $K_{i+1} = \frac{K_i[x]}{\langle g(x)\rangle}$.
4) Repeat the process for K_{i+1} until $f(x)$ completely factors.

B. Computing decoding vector in exact gradient coding

The following algorithm can be used to compute the decoding vector in the exact gradient coding scheme [11, Algorithm 2].

Algorithm 2 Computing decoding vector $\Lambda(\mathcal{K})$

Data: any vector $\mathbf{x}' \in \mathbb{C}^n$ such that $\mathbf{x}'\mathbf{B} = \beta$

Input:
A set $\mathcal{K} \subseteq [n]$ of $n - s$ non-stragglers

Output: a vector $\Lambda(\mathcal{K})$ such that $supp(\Lambda(\mathcal{K})) \subseteq \mathcal{K}$ and $\Lambda(\mathcal{K})\mathbf{B} = \beta$

find $\mathbf{f} \in \mathbb{C}^s$ such that $\mathbf{f}\mathbf{G}_{\mathcal{K}^c} = -\mathbf{x}'_{\mathcal{K}^c}\mathbf{D}^{-1}_{\mathcal{K}^c}$
$\mathbf{y} \leftarrow \mathbf{f}\mathbf{G}\mathbf{D}$
return $\Lambda(\mathcal{K}) \leftarrow \mathbf{y} + \mathbf{x}'$