




Article

Decoding Linear Codes over Chain Rings Given by Parity Check Matrices

José Gómez-Torrecillas ^{1,*}, F. J. Lobillo ² and Gabriel Navarro ³¹ Department of Algebra and IMAG, University of Granada, E18071 Granada, Spain² Department of Algebra and CITIC, University of Granada, E18071 Granada, Spain; jlobillo@ugr.es³ Department of Computer Science and Artificial Intelligence and CITIC, University of Granada, E18071 Granada, Spain; gnavarro@ugr.es

* Correspondence: gomezj@ugr.es; Tel.: +34-958-240-470

Abstract: We design a decoding algorithm for linear codes over finite chain rings given by their parity check matrices. It is assumed that decoding algorithms over the residue field are known at each degree of the adic decomposition.

Keywords: chain ring; linear code; decoding; parity check matrix



Citation: Gómez-Torrecillas, J.; Lobillo, F.J.; Navarro, G. Decoding Linear Codes over Chain Rings Given by Parity Check Matrices. *Mathematics* **2021**, *9*, 1878. <https://doi.org/10.3390/math9161878>

Academic Editor: Jon-Lark Kim

Received: 13 July 2021

Accepted: 4 August 2021

Published: 7 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

One of the first applications of linear codes whose underlying alphabet is not a finite field appears in [1], where nonlinear binary codes are built from \mathbb{Z}_4 -linear codes by means of the Gray map. Since then, considerable research efforts have focused on linear codes having a finite ring R as their alphabet. Normally, R is assumed to enjoy suitable properties. For instance, Wood, in [2], states MacWilliams identities for finite Frobenius rings, extending the foundations of coding theory to linear codes over Frobenius rings. In [3] it is proven that finite Frobenius rings are Frobenius algebras over their characteristic subrings, which enriches the duality theory for linear codes over this kind of alphabet.

Feng et al. connect linear codes over finite chain rings to network coding in [4,5] by means of matrix channels. They provide a general description of linear codes over finite chain rings, where the m -adic decomposition is made with respect to any set of representatives containing the element zero.

Concerning efficient decoding algorithms, a framework for decoding linear codes over Galois rings is proposed in [6], which generalizes previous works like [7]. This decoding framework assumes that there is a chain of linear codes over the residue field which have efficient decoding algorithms. These codes are defined by their generating matrices.

In this paper we improve the decoding framework of [6] in two ways. In Appendix A we observe that the decoding scheme from [6] works, with slight modifications, over any finite chain ring and for any set of representatives containing 0 in each degree. Anyway, the efficiently decodable codes are still ordered in a chain. In Section 3 we introduce a new framework where the codes are provided by parity check matrices. This viewpoint has an advantage: the codes over the residue field which are associated to each degree in the corresponding m -adic decomposition do not need to be ordered in a chain. We gain thus flexibility to build them from codes over fields with good decoding algorithms. In Section 4, Smith normal form of matrices over chain rings are used to compute generating matrices from parity check ones. So a complete coding/decoding scheme is provided. We have also included the Sagemath code of the proposed framework in Appendix B.

2. Preliminaries

Throughout this paper, the word ring means finite commutative ring with identity. A ring is said to be a chain ring if its ideals form a chain under inclusion. Every chain ring is a

local ring and, therefore, its elements admit “adic” expansions with respect to the maximal ideal. More precisely, let R be a finite local ring with maximal ideal \mathfrak{m} . Nakayama’s Lemma shows that the powers of \mathfrak{m} form a finite chain with strict inclusions

$$R \supset \mathfrak{m} \supset \dots \supset \mathfrak{m}^{\nu-1} \supset \mathfrak{m}^\nu = \{0\},$$

with $\mathfrak{m}^{\nu-1} \neq \{0\}$ for some positive integer ν called the nilpotency index of R . If R is a chain ring, then all its ideals appear in this chain.

Given $r \in R$, we set

$$\text{deg}(r) = \max\{i \leq \nu : r \in \mathfrak{m}^i\},$$

the degree of r . For $i = 0, \dots, \nu - 1$ we consider the canonical projection maps

$$\pi_i : \mathfrak{m}^i \rightarrow \mathfrak{m}^i / \mathfrak{m}^{i+1},$$

and we fix maps

$$\epsilon_{[i]} : \mathfrak{m}^i / \mathfrak{m}^{i+1} \rightarrow \mathfrak{m}^i,$$

such that $\pi_i \epsilon_{[i]} = \text{id}_{\mathfrak{m}^i / \mathfrak{m}^{i+1}}$.

Every $r \in R$ is expressed as

$$r = r_{[0]} + r_{[1]} + \dots + r_{[\nu-1]}, \tag{1}$$

for uniquely determined $r_{[i]} \in \text{im } \epsilon_{[i]}$, for $i = 0, \dots, \nu - 1$. This expression is referred to as the $(\mathfrak{m}, \epsilon_{[0]}, \dots, \epsilon_{[\nu-1]})$ -adic expansion of r . Indeed, if r is written as in (1), then, since $\pi_k(r_{[j]}) = 0$ whenever $j > k$, we have

$$\pi_i(r_{[i]}) = \pi_i(r - r_{[0]} - \dots - r_{[i-1]}), \tag{2}$$

for every $i = 1, \dots, \nu - 1$. Now, $r_{[i]} \in \text{im } \epsilon_{[i]}$ implies $\epsilon_{[i]} \pi_i(r_{[i]}) = r_{[i]}$, so we deduce

$$r_{[i]} = \epsilon_{[i]} \pi_i(r - r_{[0]} - \dots - r_{[i-1]}). \tag{3}$$

From (1) we also get that $\pi_0(r) = \pi_0(r_{[0]})$ and, thus,

$$r_{[0]} = \epsilon_{[0]} \pi_0(r). \tag{4}$$

Equality (4), in conjunction with the recursive formula (3), shows that the elements $r_{[i]}$ are uniquely determined by r .

This idea also shows how to compute, granting in this way the existence of the expression (1), the elements $r_{[i]}$ from a given $r \in R$. In fact, $r_{[0]}$ is computed according to (4), and the subsequent elements $r_{[1]}, \dots, r_{[\nu-1]}$ are defined recursively by (3). Observe that

$$r - r_{[0]} - \dots - r_{[i-1]} \in \mathfrak{m}^i,$$

as a consequence of a recursive application of the identities $\pi_i \epsilon_{[i]} = \text{id}_{\mathfrak{m}^i / \mathfrak{m}^{i+1}}$, (4) and (3). Finally, we may see that $\mathfrak{m}^\nu = \{0\}$ implies $\epsilon_{[\nu-1]} \pi_{\nu-1} = \text{id}_{\mathfrak{m}^{\nu-1}}$, which, by (3), gives

$$r_{[\nu-1]} = \epsilon_{[\nu-1]} \pi_{\nu-1}(r - r_{[0]} - \dots - r_{[\nu-2]}) = r - r_{[0]} - \dots - r_{[\nu-2]},$$

leading to (1).

When R is a chain ring, its maximal ideal is principal, and we may then choose $m \in R$ such that $\mathfrak{m} = Rm$ (see e.g., Proposition 2.1 in [8] or §XVII in [9]). It follows that $\mathfrak{m}^i = Rm^i$ for each $0 \leq i \leq \nu - 1$.

Taking advantage of the well known fact that $\mathfrak{m}^i / \mathfrak{m}^{i+1}$ is a vector space of dimension 1 over the residue field $F = R/\mathfrak{m}$, we obtain a bijective map

$$\{\epsilon_i : F \rightarrow R \mid \pi_0 \epsilon_i = \text{id}_F\} \rightarrow \{\epsilon_{[i]} : \mathfrak{m}^i / \mathfrak{m}^{i+1} \rightarrow \mathfrak{m}^i \mid \pi_i \epsilon_{[i]} = \text{id}_{\mathfrak{m}^i / \mathfrak{m}^{i+1}}\}$$

as follows. The multiplication map $R \xrightarrow{\cdot m^i} \mathfrak{m}^i/\mathfrak{m}^{i+1}$ induces an isomorphism of R -modules

$$\lambda_i : R/\mathfrak{m} \rightarrow \mathfrak{m}^i/\mathfrak{m}^{i+1}, \quad (r + \mathfrak{m} \mapsto rm^i + \mathfrak{m}^{i+1}).$$

In this way, for each $i = 0, 1, \dots, \nu - 1$, given $\epsilon_i : F \rightarrow R$ such that $\pi_0\epsilon_i = \text{id}_{R/\mathfrak{m}}$, we may define the map $\epsilon_{[i]} = (\cdot m^i)\epsilon_i\lambda_i^{-1}$. Now, since $\lambda_i\pi_0 = \pi_i(\cdot m^i)$, we get

$$\pi_i\epsilon_{[i]} = \pi_i(\cdot m^i)\epsilon_i\lambda_i^{-1} = \lambda_i\pi_0\epsilon_i\lambda_i^{-1} = \lambda_i \text{id}_{R/\mathfrak{m}} \lambda_i^{-1} = \text{id}_{\mathfrak{m}^i/\mathfrak{m}^{i+1}}.$$

The maps $\epsilon_{[i]}$ obey the rule

$$\epsilon_{[i]}(rm^i + \mathfrak{m}^{i+1}) = \epsilon_i(r + \mathfrak{m})m^i. \tag{5}$$

Summing up the relevant information so far obtained, we state the following proposition.

Proposition 1. *Let $F = R/\mathfrak{m}$ denote the residue field of R . Fix a generator m of \mathfrak{m} and splitting maps $\epsilon_0, \epsilon_1, \dots, \epsilon_{\nu-1} : F \rightarrow R$ such that $\pi_0\epsilon_i = \text{id}_F$ for all $0 \leq i \leq \nu - 1$. Then, for each $r \in R$, there exist uniquely determined $\rho_0, \dots, \rho_{\nu-1} \in F$ such that*

$$r = \epsilon_0(\rho_0) + \epsilon_1(\rho_1)m + \dots + \epsilon_{\nu-1}(\rho_{\nu-1})m^{\nu-1}. \tag{6}$$

Moreover, if $\epsilon_i(0) = 0$ for each $0 \leq i \leq \nu - 1$, then $r \in \mathfrak{m}^i$ if and only if $\rho_0 = \dots = \rho_{i-1} = 0$.

Proof. Define the maps $\epsilon_{[i]}$ according to (5). For each $i = 0, \dots, \nu - 1$ set, in the unique decomposition (1), $r_{[i]} = \epsilon_i(\rho_i)m^i$ for suitable $\rho_i \in F$. Indeed, from this last equality we see that $\pi_i(r_{[i]}) = \lambda_i(\rho_i)$, which proves that ρ_i is uniquely determined by $r_{[i]}$. We thus obtain the expansion (6).

Now, from (2) we derive

$$\rho_i = \lambda_i^{-1}(r - \epsilon_0(\rho_0) - \epsilon_1(\rho_1)m - \dots - \epsilon_{i-1}(\rho_{i-1})m^{i-1} + \mathfrak{m}^{i+1}), \tag{7}$$

with $\rho_0 = \epsilon_0(r + \mathfrak{m})$. Using recursively (7), we prove that, when $\epsilon_i(0) = 0$, $r \in \mathfrak{m}^i$ if and only if $\rho_0 = \dots = \rho_{i-1} = 0$. \square

Remark 1. *The coefficients ρ_i can be computed recursively from (7).*

The decomposition (6) depends on $\epsilon_0, \epsilon_1, \dots, \epsilon_{\nu-1}$ and m . We call it the $(m, \epsilon_0, \epsilon_1, \dots, \epsilon_{\nu-1})$ -adic decomposition of r , or m -adic decomposition, when no ambiguity is expected. If $\epsilon = \epsilon_0 = \epsilon_1 = \dots = \epsilon_{\nu-1}$ and $\epsilon(0) = 0$, this decomposition coincides with that of Equation (2) in [4], since ϵ gives a choice of residuals modulo m .

Definition 1. *A tuple $(m, \epsilon_0, \dots, \epsilon_{\nu-1})$ such that $\mathfrak{m} = Rm$ is called a splitting structure for R if, for each $0 \leq i \leq \nu - 1$, $\epsilon_i : F \rightarrow R$ satisfies $\pi_0\epsilon_i = \text{id}_F$ and $\epsilon_i(0) = 0$,*

Remark 2. *Let $(m, \epsilon_0, \dots, \epsilon_{\nu-1})$ be a splitting structure for R . Assume $r = um^i$ for some $u \in R$ and let (6) be its m -adic decomposition. Then, by (7) and Proposition 1,*

$$\rho_i = \lambda_i^{-1}(r + \mathfrak{m}^{i+1}) = \lambda_i^{-1}(um^i + \mathfrak{m}^{i+1}) = u + \mathfrak{m} = \pi_0(u).$$

Hence, ρ_i does not depend on the splitting structure.

The m -adic expansion (6) is extended to matrices in a straightforward way. Let $A^{s \times t}$ denote the set of all matrices of size $s \times t$ with coefficients in a commutative ring A , which

is a free A -module. We may extend any map $\epsilon : F \rightarrow R$ component-wise to a map $\epsilon : F^{s \times t} \rightarrow R^{s \times t}$. Then, every matrix $L \in R^{s \times t}$ has an m -adic expansion

$$L = \epsilon_0(\Lambda_0) + \epsilon_1(\Lambda_1)m + \dots + \epsilon_{\nu-1}(\Lambda_{\nu-1})m^{\nu-1},$$

for uniquely determined matrices $\Lambda_i \in F^{s \times t}$.

Although the splitting maps are not additive neither multiplicative, they obey some relations which will be used. Concretely, let $\rho, \sigma \in F$ and $\epsilon_i, \epsilon_j, \epsilon_k : F \rightarrow R$ splittings. Since $\pi_0 : R \rightarrow F$ is a ring morphism, it follows that

$$\pi_0(\epsilon_i(\rho + \sigma) - \epsilon_j(\rho) - \epsilon_k(\sigma)) = (\rho + \sigma) - \rho - \sigma = 0$$

and

$$\pi_0(\epsilon_i(\rho\sigma) - \epsilon_j(\rho)\epsilon_k(\sigma)) = (\rho\sigma) - \rho\sigma = 0,$$

hence

$$\begin{aligned} \epsilon_i(\rho + \sigma) - \epsilon_j(\rho) - \epsilon_k(\sigma) &\in \mathfrak{m}, \\ \epsilon_i(\rho\sigma) - \epsilon_j(\rho)\epsilon_k(\sigma) &\in \mathfrak{m}. \end{aligned} \tag{8}$$

The structure of the finite chain rings is well known, see (XVII.5) Theorem in [9]. We will not use this description in full generality, so we only recall the rings that appear in our examples.

Example 1. Recall that a Galois ring $GR(p^\alpha, \beta)$ is an extension of degree β of $\mathbb{Z}/\mathbb{Z}p^\alpha$, i.e., $GR(p^\alpha, \beta) \cong (\mathbb{Z}/\mathbb{Z}p^\alpha)[x]/\langle f \rangle$, where f is a basic monic irreducible polynomial in $(\mathbb{Z}/\mathbb{Z}p^\alpha)[x]$ of degree β . Its maximal ideal is generated by the prime p . The nilpotency index of p is α , and $F \cong (\mathbb{Z}/\mathbb{Z}p)[x]/\langle \bar{f} \rangle$, where \bar{f} is the canonical projection of f to $(\mathbb{Z}/\mathbb{Z}p)[x]$. In particular, $\mathbb{Z}/\mathbb{Z}p^\alpha = GR(p^\alpha, 1)$ is a chain ring. On the other side, $GF(p^\beta) = GR(p, \beta)$ is a field, so it is trivially a chain ring.

Example 2. The ring $\mathbb{F}_{p^\alpha}[x]/\langle x^\beta \rangle$, where \mathbb{F}_{p^α} is the field with p^α elements, is also a chain ring. The maximal ideal is generated by x , and it has nilpotency index β . Of course $F \cong \mathbb{F}_{p^\alpha}$.

In the rest of the paper, \mathcal{C} is an R -linear code of length n . Vectors are represented by boldface letters, whilst matrices with uppercase letters. We use Latin alphabet to represent elements in R and greek alphabet to represent elements of F . Moreover, given a matrix M over R with n rows, we denote

$$\text{im}(M) = \{\mathbf{x}M \mid \mathbf{x} \in R^n\} \text{ and } \text{ker}(M) = \{\mathbf{x} \in R^n \mid \mathbf{x}M = 0\}.$$

The same applies to matrices over the residue field F .

Remark 3. By an abuse of language, for $\mathbf{v} = (v_0, \dots, v_{n-1}) \in R^n$, we say that $\mathbf{v} \in \mathfrak{m}^i$ if $v_j \in \mathfrak{m}^i$ for all $0 \leq j \leq n - 1$. The same convention applies to matrices.

3. Decoding via Parity Check

Let us fix a chain ring R with maximal ideal \mathfrak{m} and nilpotency index ν , i.e., $\mathfrak{m}^\nu = \{0\}$ and $\mathfrak{m}^{\nu-1} \neq \{0\}$. We also fix a splitting structure $(m, \epsilon_0, \dots, \epsilon_{\nu-1})$ for R . There are two standard ways to present an R -linear code \mathcal{C} , as the image $\mathcal{C} = \text{im}(G)$ of a generating matrix G or as the kernel $\mathcal{C} = \text{ker}(H)$ of a parity check matrix H . In the first case, by §IV in [5], we do not lose generality if we assume $\mathcal{C} = \text{im}(G)$, where

$$G = \begin{pmatrix} G^{(0)} \\ G^{(1)} \\ \vdots \\ G^{(\nu-1)} \end{pmatrix}$$

and, for each $0 \leq i \leq \nu - 1$, $G^{(i)}$ is a $k_i \times n$ matrix whose m -adic decomposition is

$$G^{(i)} = \epsilon_i(\Gamma_i^{(i)})m^i + \epsilon_{i+1}(\Gamma_{i+1}^{(i)})m^{i+1} + \dots + \epsilon_{\nu-1}(\Gamma_{\nu-1}^{(i)})m^{\nu-1},$$

with matrices $\Gamma_j^{(i)}$ whose entries are in F , and $\Gamma_i^{(i)}$ is full rank.

The decoding framework introduced in [6] and expounded in Appendix A uses this presentation of codes as images. It needs a chain of linear codes over the residue field F

$$\text{im}(\Gamma_0^{(0)}) \subseteq \text{im} \begin{pmatrix} \Gamma_1^{(1)} \\ \Gamma_0^{(0)} \end{pmatrix} \subseteq \dots \subseteq \text{im} \begin{pmatrix} \Gamma_{\nu-1}^{(\nu-1)} \\ \vdots \\ \Gamma_0^{(0)} \end{pmatrix}$$

with efficient decoding algorithms. There are several ways to get it. For instance, even with classical linear codes, if we want to use BCH (Bose-Chaudhuri-Hocquenghem) codes, we shall use a decreasing chain of defining sets to build the chain of codes. Goppa codes can also be used but taking as the Goppa polynomial of one code in the chain a divisor of the Goppa polynomial of the previous code. Anyway, this is a limitation of the possible codes we can use at each degree.

We are interested in the second presentation, so let \mathcal{C} be an R -linear code given by a parity check matrix $H \in R^{n \times q}$, i.e.,

$$\mathcal{C} = \ker(H) = \{\mathbf{x} \in R^n \mid \mathbf{x}H = 0\}.$$

In this section, we develop a new decoding framework based on syndrome decoding for each degree, i.e., we use the parity check matrices and the syndromes of the received words to decode. This strategy allows one to choose independently the linear codes over the residue field at each degree.

By §II.D in [4], we can replace H by its column reduced canonical form, so we do not lose generality if we assume

$$H = \left(H^{(0)} \mid H^{(1)} \mid \dots \mid H^{(\nu-1)} \right), \tag{9}$$

where $H^{(i)} = \sum_{j=i}^{\nu-1} \epsilon_j(\Theta_j^{(i)})m^j$

for suitable matrices $\Theta_j^{(i)} \in F^{n \times q_i}$ such that the matrices $\Theta_i^{(i)}$ are full rank.

As usual, let $\mathbf{y} = \mathbf{c} + \mathbf{e}$, where $\mathbf{c} \in \mathcal{C}$ and \mathbf{e} is the error vector. The syndrome is

$$\mathbf{s} = \mathbf{y}H = \mathbf{e}H,$$

and we denote

$$\mathbf{s}^{(i)} = \mathbf{y}H^{(i)} = \mathbf{e}H^{(i)}, \quad 0 \leq i \leq \nu - 1.$$

Our decoding framework computes \mathbf{e} from H and \mathbf{s} by means of an iterative process. Let us introduce notation for the corresponding m -adic expansions:

$$\mathbf{e} = \sum_{l=0}^{\nu-1} \epsilon_l(\boldsymbol{\xi}_l)m^l, \tag{10}$$

$$\mathbf{s}^{(i)} = \sum_{j=i}^{\nu-1} \epsilon_j(\boldsymbol{\sigma}_j^{(i)})m^j, \quad 0 \leq i \leq \nu - 1.$$

We have taken into account that $s^{(i)} \in \mathfrak{m}^i$. Observe that, for each $0 \leq i \leq \nu - 1$,

$$\begin{aligned} \mathbf{e}H^{(i)} &= \sum_{l_0=0}^{\nu-1} \epsilon_{l_0}(\boldsymbol{\xi}_{l_0})m^{l_0} \sum_{j_0=i}^{\nu-1} \epsilon_{j_0}(\Theta_{j_0}^{(i)})m^{j_0} \\ &= \sum_{j_0=i}^{\nu-1} \sum_{l_0=0}^{\nu-1} \epsilon_{l_0}(\boldsymbol{\xi}_{l_0})\epsilon_{j_0}(\Theta_{j_0}^{(i)})m^{j_0+l_0} \\ &= \sum_{j=i}^{\nu-1} \sum_{l=0}^{j-i} \epsilon_l(\boldsymbol{\xi}_l)\epsilon_{j-l}(\Theta_{j-l}^{(i)})m^j, \end{aligned} \tag{11}$$

where we use that $m^\nu = 0$ and we performed the change of variable $j = j_0 + l_0$. Hence,

$$\sum_{j=i}^{\nu-1} \epsilon_j(\boldsymbol{\sigma}_j^{(i)})m^j = \sum_{j=i}^{\nu-1} \sum_{l=0}^{j-i} \epsilon_l(\boldsymbol{\xi}_l)\epsilon_{j-l}(\Theta_{j-l}^{(i)})m^j \tag{12}$$

for each $0 \leq i \leq \nu - 1$. The right hand side of (12) needs not to be an m -adic decomposition, so we cannot infer from (12) any equality of the corresponding coefficients of each m^j .

Let us describe the iterative decoding framework.

3.1. First Step: Computing $\boldsymbol{\xi}_0$

Equation (12), when $i = \nu - 1$, gives

$$\epsilon_{\nu-1}(\boldsymbol{\sigma}_{\nu-1}^{(\nu-1)})m^{\nu-1} = \epsilon_0(\boldsymbol{\xi}_0)\epsilon_{\nu-1}(\Theta_{\nu-1}^{(\nu-1)})m^{\nu-1}.$$

By (8),

$$\epsilon_0(\boldsymbol{\xi}_0)\epsilon_{\nu-1}(\Theta_{\nu-1}^{(\nu-1)}) - \epsilon_{\nu-1}(\boldsymbol{\xi}_0\Theta_{\nu-1}^{(\nu-1)}) \in \mathfrak{m},$$

hence

$$\epsilon_0(\boldsymbol{\xi}_0)\epsilon_{\nu-1}(\Theta_{\nu-1}^{(\nu-1)})m^{\nu-1} = \epsilon_{\nu-1}(\boldsymbol{\xi}_0\Theta_{\nu-1}^{(\nu-1)})m^{\nu-1}.$$

By Proposition 1,

$$\boldsymbol{\sigma}_{\nu-1}^{(\nu-1)} = \boldsymbol{\xi}_0\Theta_{\nu-1}^{(\nu-1)}. \tag{13}$$

Proposition 2. Let $\mathfrak{d}_{\nu-1}$ be a decoding algorithm for the linear code $\ker(\Theta_{\nu-1}^{(\nu-1)})$. If the weight of $\boldsymbol{\xi}_0$ is below the correction capability of $\mathfrak{d}_{\nu-1}$, then $\boldsymbol{\xi}_0 = \mathfrak{d}_{\nu-1}(\boldsymbol{\sigma}_{\nu-1}^{(\nu-1)})$.

Proof. Just observe that $\boldsymbol{\xi}_0$ is the only one solution of (13) whose weight is below the correction capability of $\ker(\Theta_{\nu-1}^{(\nu-1)})$. \square

3.2. Second Step: Computing $\boldsymbol{\xi}_1$

We include this second step to help the reader to follow the framework. At this step, $\boldsymbol{\xi}_0$ is known. If we put $i = \nu - 2$ in (12), we have

$$\begin{aligned} \epsilon_{\nu-2}(\boldsymbol{\sigma}_{\nu-2}^{(\nu-2)})m^{\nu-2} + \epsilon_{\nu-1}(\boldsymbol{\sigma}_{\nu-1}^{(\nu-2)})m^{\nu-1} = \\ \epsilon_0(\boldsymbol{\xi}_0)\epsilon_{\nu-2}(\Theta_{\nu-2}^{(\nu-2)})m^{\nu-2} + \epsilon_0(\boldsymbol{\xi}_0)\epsilon_{\nu-1}(\Theta_{\nu-1}^{(\nu-2)})m^{\nu-1} + \epsilon_1(\boldsymbol{\xi}_1)\epsilon_{\nu-2}(\Theta_{\nu-2}^{(\nu-2)})m^{\nu-1}. \end{aligned} \tag{14}$$

Since the vector $\boldsymbol{\xi}_0$ is assumed to be known, the element

$$\begin{aligned} \epsilon_{\nu-2}(\boldsymbol{\sigma}_{\nu-2}^{(\nu-2)})m^{\nu-2} + \epsilon_{\nu-1}(\boldsymbol{\sigma}_{\nu-1}^{(\nu-2)})m^{\nu-1} - \\ \epsilon_0(\boldsymbol{\xi}_0)\epsilon_{\nu-2}(\Theta_{\nu-2}^{(\nu-2)})m^{\nu-2} - \epsilon_0(\boldsymbol{\xi}_0)\epsilon_{\nu-1}(\Theta_{\nu-1}^{(\nu-2)})m^{\nu-1} \in \mathfrak{m}^{\nu-1} \end{aligned}$$

can be computed. Hence, by Proposition 1, there exists $\delta \in F^n$ such that

$$\begin{aligned} \epsilon_{\nu-2}(\sigma_{\nu-2}^{(\nu-2)})m^{\nu-2} + \epsilon_{\nu-1}(\sigma_{\nu-1}^{(\nu-2)})m^{\nu-1} - \\ \epsilon_0(\xi_0)\epsilon_{\nu-2}(\Theta_{\nu-2}^{(\nu-2)})m^{\nu-2} - \epsilon_0(\xi_0)\epsilon_{\nu-1}(\Theta_{\nu-1}^{(\nu-2)})m^{\nu-1} = \epsilon_{\nu-1}(\delta)m^{\nu-1}, \end{aligned} \tag{15}$$

which can be computed since all elements appearing in its definition are known. Equations (14) and (15) imply

$$\epsilon_{\nu-1}(\delta)m^{\nu-1} = \epsilon_1(\xi_1)\epsilon_{\nu-2}(\Theta_{\nu-2}^{(\nu-2)})m^{\nu-1}.$$

By (8),

$$\epsilon_1(\xi_1)\epsilon_{\nu-2}(\Theta_{\nu-2}^{(\nu-2)}) - \epsilon_{\nu-1}(\xi_1\Theta_{\nu-2}^{(\nu-2)}) \in \mathfrak{m},$$

hence

$$\epsilon_1(\xi_1)\epsilon_{\nu-2}(\Theta_{\nu-2}^{(\nu-2)})m^{\nu-1} = \epsilon_{\nu-1}(\xi_1\Theta_{\nu-2}^{(\nu-2)})m^{\nu-1}.$$

By Proposition 1, it follows that

$$\delta = \xi_1\Theta_{\nu-2}^{(\nu-2)}. \tag{16}$$

Proposition 3. Let $\mathfrak{d}_{\nu-2}$ be a decoding algorithm for the linear code $\ker(\Theta_{\nu-2}^{(\nu-2)})$. If the weight of ξ_1 is below the correction capability of $\mathfrak{d}_{\nu-2}$, then $\xi_1 = \mathfrak{d}_{\nu-2}(\delta)$.

Proof. Same proof that Proposition 2, ξ_1 is the only one solution of (16) whose weight is below the correction capability of $\ker(\Theta_{\nu-2}^{(\nu-2)})$. \square

3.3. General Step: Computing ξ_l

We assume that ξ_0, \dots, ξ_{l-1} have been computed. Proceeding as in the previous cases, Equation (12) for $i = \nu - 1 - l$ provides

$$\begin{aligned} \sum_{j=\nu-1-l}^{\nu-1} \epsilon_j(\sigma_j^{(\nu-1-l)})m^j &= \sum_{j=\nu-1-l}^{\nu-1} \sum_{i=0}^{j-\nu+1+l} \epsilon_i(\xi_i)\epsilon_{j-i}(\Theta_{j-i}^{(\nu-1-l)})m^j \\ &= \sum_{j=\nu-1-l}^{\nu-2} \sum_{i=0}^{j-\nu+1+l} \epsilon_i(\xi_i)\epsilon_{j-i}(\Theta_{j-i}^{(\nu-1-l)})m^j \\ &\quad + \sum_{i=0}^l \epsilon_i(\xi_i)\epsilon_{\nu-1-i}(\Theta_{\nu-1-i}^{(\nu-1-l)})m^{\nu-1} \\ &= \sum_{j=\nu-1-l}^{\nu-2} \sum_{i=0}^{j-\nu+1+l} \epsilon_i(\xi_i)\epsilon_{j-i}(\Theta_{j-i}^{(\nu-1-l)})m^j \\ &\quad + \sum_{i=0}^{l-1} \epsilon_i(\xi_i)\epsilon_{\nu-1-i}(\Theta_{\nu-1-i}^{(\nu-1-l)})m^{\nu-1} \\ &\quad + \epsilon_l(\xi_l)\epsilon_{\nu-1-l}(\Theta_{\nu-1-l}^{(\nu-1-l)})m^{\nu-1}. \end{aligned} \tag{17}$$

By Proposition 1, there exists $\delta \in F^n$ such that

$$\begin{aligned} \sum_{j=\nu-1-l}^{\nu-1} \epsilon_j(\sigma_j^{(\nu-1-l)})m^j - \sum_{j=\nu-1-l}^{\nu-2} \sum_{i=0}^{j-\nu+1+l} \epsilon_i(\xi_i)\epsilon_{j-i}(\Theta_{j-i}^{(\nu-1-l)})m^j \\ - \sum_{i=0}^{l-1} \epsilon_i(\xi_i)\epsilon_{\nu-1-i}(\Theta_{\nu-1-i}^{(\nu-1-l)})m^{\nu-1} = \epsilon_{\nu-1}(\delta)m^{\nu-1} \in \mathfrak{m}^{\nu-1}. \end{aligned} \tag{18}$$

The left hand side of the equality in (18) is known because $i < l$ in all summands. So vector δ can be computed. By (8),

$$\epsilon_l(\xi_l)\epsilon_{v-1-l}(\Theta_{v-1-l}^{(v-1-l)}) - \epsilon_{v-1}(\xi_l\Theta_{v-1-l}^{(v-1-l)}) \in \mathfrak{m},$$

hence

$$\epsilon_l(\xi_l)\epsilon_{v-1-l}(\Theta_{v-1-l}^{(v-1-l)})m^{v-1} = \epsilon_{v-1}(\xi_l\Theta_{v-1-l}^{(v-1-l)})m^{v-1}. \tag{19}$$

Therefore, Equations (17)–(19) imply

$$\epsilon_{v-1}(\delta)m^{v-1} = \epsilon_{v-1}(\xi_l\Theta_{v-1-l}^{(v-1-l)})m^{v-1}.$$

It follows, by Proposition 1 again, that

$$\delta = \xi_l\Theta_{v-1-l}^{(v-1-l)}. \tag{20}$$

Proposition 4. Let \mathfrak{d}_{v-1-l} be a decoding algorithm for the linear code $\ker(\Theta_{v-1-l}^{(v-1-l)})$. If the weight of ξ_l is below the correction capability of \mathfrak{d}_{v-1-l} , then $\xi_l = \mathfrak{d}_{v-1-l}(\delta)$.

Proof. As we observed before in Propositions 2 and 3, ξ_l is the unique solution of (20) whose weight is below the correction capability of $\ker(\Theta_{v-1-l}^{(v-1-l)})$. \square

The decoding framework is summarized in Algorithm 1.

Algorithm 1: Syndrome decoding.

Parameters H given as in (11), and decoding algorithms $\mathfrak{d}_0, \mathfrak{d}_1, \dots, \mathfrak{d}_{v-1}$

Input $\mathbf{s} = (\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \dots, \mathbf{s}^{(v-1)}) \in R^q$

Output The error vector $\mathbf{e} = \sum_{l=0}^{v-1} \epsilon_l(\xi_l)m^l$.

Assumption For each $0 \leq l \leq v-1$, the Hamming weight of ξ_l is below the correction capability of \mathfrak{d}_l .

- 1: For each $0 \leq i \leq v-1$, compute the m -adic expansion of $\mathbf{s}^{(i)}$ in (10)
 - 2: **for** $0 \leq l \leq v-1$ **do**
 - 3: Compute δ from (18)
 - 4: $\xi_l = \mathfrak{d}_l(\delta)$
 - 5: **end for**
 - 6: **return** $\sum_{l=0}^{v-1} \epsilon_l(\xi_l)m^l$.
-

Theorem 1. Let $\mathbf{y} \in R^n$ and $\mathbf{s}^{(i)} = \mathbf{y}H^{(i)}$ for each $0 \leq i \leq v-1$. If there exists $\mathbf{e} = \sum_{l=0}^{v-1} \epsilon_l(\xi_l)m^l \in R^n$ such that $(\mathbf{y} - \mathbf{e})H = 0$ and the weight of ξ_l is below the correction capability of \mathfrak{d}_l for each $0 \leq l \leq v-1$, then Algorithm 1 correctly computes it.

Proof. Follows directly from Propositions 2–4. \square

Example 3. In order to explain how this decoding framework works, we are going to develop a step by step example. Let $R = GR(4, 2)$, so $F = \mathbb{F}_4 = \mathbb{F}_2[a] / \langle a^2 + a + 1 \rangle$. Let $C = \ker(H)$, where

$$H = \left(\begin{array}{cc|cc} 2a + 3 & 2a + 2 & 2 & 0 \\ 2a + 2 & 2a + 1 & 0 & 2 \\ 2a + 1 & 2a + 3 & 2 & 2 \\ & 3 & 3a + 2 & 2a \\ & 1 & 3a + 1 & 2a + 2 \end{array} \right).$$

The splitting structure is $(2, \epsilon_0, \epsilon_1)$, where

	0	1	a	$a + 1$
ϵ_0	0	$2a + 1$	$3a + 2$	$a + 3$
ϵ_1	0	3	$3a$	$3a + 1$

According to this splitting structure, we have

$$H^{(0)} = \begin{pmatrix} 2a + 3 & 2a + 2 \\ 2a + 2 & 2a + 1 \\ 2a + 1 & 2a + 3 \\ 3 & 3a + 2 \\ 1 & 3a + 1 \end{pmatrix} = \epsilon_0 \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & a \\ 1 & a + 1 \end{pmatrix} + 2\epsilon_1 \begin{pmatrix} 1 & a + 1 \\ a + 1 & 0 \\ 0 & 1 \\ a + 1 & 0 \\ a & a + 1 \end{pmatrix}$$

and

$$H^{(1)} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 2 & 2 \\ 2 & 2a \\ 2 & 2a + 2 \end{pmatrix} = 2\epsilon_1 \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & a \\ 1 & a + 1 \end{pmatrix}.$$

Observe that $\Theta_0^{(0)}$ and $\Theta_1^{(1)}$ are the parity check matrices of a Hamming code of length 5 and dimension 3, which corrects one single error. It can be checked that

$$(2, 2a + 1, a + 3, 2a, 3a + 3) \in \mathcal{C}.$$

Although the error vector

$$(2a + 2, 0, 0, 3a + 2, 0)$$

has Hamming weight 2, its $(2, \epsilon_0, \epsilon_1)$ -adic decomposition is

$$(2a + 2, 0, 0, 3a + 2, 0) = \epsilon_0(0, 0, 0, a, 0) + 2\epsilon_1(a + 1, 0, 0, 0, 0),$$

so our framework should be able to decode the received word

$$\begin{aligned} (2, 2a + 1, a + 3, 2a, 3a + 3) + (2a + 2, 0, 0, 3a + 2, 0) \\ = (2a, 2a + 1, a + 3, a + 2, 3a + 3). \end{aligned}$$

The syndrome of the received word is

$$\begin{aligned} (2a, 2a + 1, a + 3, a + 2, 3a + 3) \left(\begin{array}{cc|cc} 2a + 3 & 2a + 2 & 2 & 0 \\ 2a + 2 & 2a + 1 & 0 & 2 \\ 2a + 1 & 2a + 3 & 2 & 2 \\ 3 & 3a + 2 & 2 & 2a \\ 1 & 3a + 1 & 2 & 2a + 2 \end{array} \right) \\ = (3a, 3a + 3, 2a, 2a + 2) = ((3a, 3a + 3), (2a, 2a + 2)) \end{aligned}$$

whose $(2, \epsilon_0, \epsilon_1)$ -decompositions are

$$(3a, 3a + 3) = \epsilon_0(a, a + 1) + 2\epsilon_1(1, a)$$

$$(2a, 2a + 2) = 2\epsilon_1(a, a + 1)$$

Algorithm 1 can now be applied. In the first round, $i = 0$, we have $\delta = \sigma_1^{(1)} = (a, a + 1)$, which is a times the fourth row of $\Theta_1^{(1)}$. So $\xi_0 = (0, 0, 0, a, 0)$.

In the second round, $i = 1$, we need to compute the $(2, \epsilon_0, \epsilon_1)$ -adic decomposition of

$$\epsilon_0(a, a + 1) + 2\epsilon_1(1, a) - \epsilon_0(0, 0, 0, a, 0)\epsilon_0 \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & a \\ 1 & a + 1 \end{pmatrix} - 2\epsilon_0(0, 0, 0, a, 0)\epsilon_1 \begin{pmatrix} 1 & a + 1 \\ a + 1 & 0 \\ 0 & 1 \\ a + 1 & 0 \\ a & a + 1 \end{pmatrix} = (2a + 2, 0),$$

which is

$$(2a + 2, 0) = 2\epsilon_1(a + 1, 0).$$

So, $\delta = (a + 1, 0)$, the first row of $\Theta_0^{(0)}$ multiplied by $a + 1$. Therefore, $\xi_1 = (a + 1, 0, 0, 0, 0)$. It follows

$$\epsilon_0(0, 0, 0, a, 0) + 2\epsilon_1(a + 1, 0, 0, 0, 0) = (2a + 2, 0, 0, 3a + 2, 0),$$

as expected.

Unlike the former case, if we make use of a different splitting structure to decode, the framework could not work. For instance, consider the splitting structure $(2, \epsilon'_0, \epsilon'_1)$, where

	0	1	a	$a + 1$
ϵ'_0	0	$2a + 3$	$3a$	$a + 1$
ϵ'_1	0	$2a + 1$	$a + 2$	$3a + 1$

The $(2, \epsilon'_0, \epsilon'_1)$ -decomposition of the error vector is

$$(2a + 2, 0, 0, 3a + 2, 0) = \epsilon'_0(0, 0, 0, a, 0) + 2\epsilon'_1(a + 1, 0, 0, 1, 0),$$

which provides a vector of Hamming weight 2 in degree 1, so Algorithm 1 does not apply.

Remark 4. Our parity check decoding framework could be used to design a McEliece like cryptosystem following the proposal in [10]. However, by Remark 2, given H as in (9), the matrices $\Theta_0^{(0)}, \dots, \Theta_{v-1}^{(v-1)}$ do not depend on the splitting structure, so an eavesdropper could use any structure to compute the parity check matrices of the linear codes over F . Therefore, the security of this possible cryptosystem would be equivalent to v consecutive linear codes over the residue field F .

Remark 5. The time complexity of Algorithm 1 is bounded by the theoretical efficiency of the chosen decoding algorithms $\mathfrak{d}_0, \dots, \mathfrak{d}_{v-1}$ (Line 4), and the intermediate calculations are described in Lines 1 and 3. Indeed, with respect to the number of elementary operations (additions, multiplications, and map images) over the residue field, let us assume that \mathfrak{d}_i belongs to $O(f_i)$ for $i = 0, \dots, v - 1$ with respect to the length of the code. Moreover, for simplicity, assume that $f_i \in O(f)$ for all i .

According to (7), an m -adic expansion can be computed by $2v$ operations, so that Line 1 belongs to $O(v^2)$. Now, the calculation of δ in Line 4 is obtained by solving a linear system over the residue field F . This can be computed by Gaussian elimination, which can be done in $O(t^\omega)$, where ω is the matrix multiplication exponent and t is the dimension of the matrix. We may consider the classical algorithm and set $\omega = 3$, so that Line 3 in each iteration of the loop belongs to $O(n^3 + f(n))$, where n is the length of the code. Thus Algorithm 1 can be executed in $O(v^2 + vn^3 + vf(n))$. In general, since $v \ll n$, we may say that the complexity belongs to $O(\max(n^3, f(n)))$.

4. Parity Check and Encoders

There are known interesting applications of linear codes over finite chain rings as those mentioned in [4,5]. So, even though the decoding framework is based on the syndrome decoding by means of parity check matrices, it is needed to provide an encoding process. So we need to build a generating matrix from the parity check matrix H which defines our code. This task may be performed by using the Smith normal form. Recall that a $k \times n$ matrix M over an arbitrary ring has a Smith normal form if there exist invertible $k \times k$ and $n \times n$ matrices P, Q and a diagonal (non necessarily square) matrix D , where $d_1, \dots, d_{\min(k,n)}$ are the elements in the main diagonal, such that $PMQ = D$ and $d_i \mid d_{i+1}$.

Any matrix over a commutative principal ideal ring has a Smith normal form (Chapter 15 in [11]). In the particular case of a chain ring, Algorithm 2 gives a way to compute this normal form which simplifies the general procedure in [11].

We may assume $k \leq n$, otherwise we can compute it for its transpose M^T , and if $PM^TQ = D$ we have $Q^T M P^T = D^T$.

Algorithm 2: Smith normal form for finite chain rings.

Input A matrix $M \in R^{k \times n}$ over a finite chain ring R with nilpotency index ν , such that $k \leq n$.

Output D, P, Q such that D is a Smith normal form, P and Q are invertible, and $D = PMQ$.

```

1: if  $k = 1$  then
2:    $Q \leftarrow I_{n \times n}$ .
3:   Find  $m_{1,j}$  of lowest degree in  $M$ .
4:   Swap columns 1 and  $j$  in  $M$  and  $Q$ 
5:   for  $2 \leq j \leq n$  do
6:     Compute  $t \in R$  such that  $m_{1,j} = tm_{1,1}$ 
7:     In  $M$  and  $Q$ , replace column  $j$  with column  $j$  minus  $t$  times column 1.
8:   end for
9:   return  $D = (m_{1,1}, 0, \dots, 0)$ ,  $P = 1$ ,  $Q$ .
10: else
11:    $S, T \leftarrow I_{k \times k}, I_{n \times n}$ 
12:   Find  $m_{i,j}$  of lowest degree in  $M$ 
13:   Swap row 1 and row  $i$  in  $M$  and  $S$ 
14:   Swap column 1 and column  $j$  in  $M$  and  $T$ 
15:   for  $2 \leq i \leq k$  do
16:     Compute  $t \in R$  such that  $m_{i,1} = tm_{1,1}$ 
17:     In  $M$  and  $S$ , replace row  $i$  with row  $i$  minus  $t$  times row 1.
18:   end for
19:   for  $2 \leq j \leq n$  do
20:     Compute  $t \in R$  such that  $m_{1,j} = tm_{1,1}$ 
21:     In  $M$  and  $T$ , replace column  $j$  with column  $j$  minus  $t$  times column 1.
22:   end for
23:   Set  $M' \in R^{(k-1) \times (n-1)}$  the matrix obtained from  $M$  deleting its first row and column.
24:   Apply Algorithm 2 to  $M'$  and compute  $D', P', Q'$  such that  $D'$  is a Smith normal form for  $M'$  and  $D' = P'M'Q'$ .
25:   return  $\left( \begin{array}{c|c} m_{1,1} & \\ \hline & D' \end{array} \right)$ ,  $\left( \begin{array}{c|c} 1 & \\ \hline & P' \end{array} \right) S$  and  $T \left( \begin{array}{c|c} 1 & \\ \hline & Q' \end{array} \right)$ 
26: end if

```

Remark 6. Observe that, once a generator m has been fixed for the maximal ideal \mathfrak{m} of R , it is easy to check that $\deg(r) = d$ if and only if $r = um^d$ where $u \in R \setminus \mathfrak{m}$, the set of units of R .

Therefore, $\deg(r) = d$ and $m^d = Rr$ are equivalent conditions on r . Since $\deg(r) \leq \deg(s)$ implies $s \in m^{\deg(r)}$, we get

$$\deg(r) \leq \deg(s) \text{ if and only if } s = tr \text{ for some } t \in R. \tag{21}$$

Since $\deg(r + s), \deg(rs) \geq \min\{\deg(r), \deg(s)\}$ for all $r, s \in R$, once $m_{1,1}$ is an element of lowest degree, any operation involving sums and products cannot decrease the degree, so, by (21), we can compute $t \in R$ in lines 6, 16 and 20, and all entries of M' have degree greater or equal than the degree of $m_{1,1}$. Since S and T are built to obtain

$$SMT = \left(\begin{array}{c|c} m_{1,1} & \\ \hline & M' \end{array} \right)$$

it follows

$$\begin{aligned} \left(\begin{array}{c|c} 1 & \\ \hline & P' \end{array} \right) SMT \left(\begin{array}{c|c} 1 & \\ \hline & Q' \end{array} \right) &= \left(\begin{array}{c|c} 1 & \\ \hline & P' \end{array} \right) \left(\begin{array}{c|c} m_{1,1} & \\ \hline & M' \end{array} \right) \left(\begin{array}{c|c} 1 & \\ \hline & Q' \end{array} \right) \\ &= \left(\begin{array}{c|c} m_{1,1} & \\ \hline & P'M'Q' \end{array} \right) \left(\begin{array}{c|c} m_{1,1} & \\ \hline & D' \end{array} \right), \end{aligned}$$

so the output of Algorithm 2 is correct.

Once the Smith normal form of the parity check matrix has been found, we may compute a generating matrix. Indeed, assume $H \in R^{n \times q}$ and let D, P, Q matrices such that D is a Smith normal form for H and $D = PHQ$. Since Q is invertible, it follows that $\mathbf{c} \in \ker(H)$ if and only if $\mathbf{c} \in \ker(HQ)$. Moreover, $\mathbf{x} \in \ker(D)$ if and only if $\mathbf{x}P \in \ker(HQ)$, so, if $\ker(D) = \text{im}(E)$ we get $\ker(H) = \text{im}(EP)$. Now, recall D is diagonal with $d_1 \mid d_2 \mid \dots \mid d_{\min\{n,q\}}$ which, by (21), is equivalent to say that $\deg(d_1) \leq \deg(d_2) \leq \dots \leq \deg(d_{\min\{n,q\}})$. It follows that E can be taken as an $n \times n$ diagonal matrix whose diagonal elements are $\{m^{v-\deg(d_1)}, m^{v-\deg(d_2)}, \dots, m^{v-\deg(d_{\min\{n,q\}})}, 1, \dots, 1\}$. We may summarize these ideas in Algorithm 3.

Algorithm 3: Generating matrix computation from a parity check matrix.

Input A parity check matrix $H \in R^{n \times q}$.

Output A matrix G such that $\ker(H) = \text{im}(G)$.

- 1: Compute D, P, Q by means of Algorithm 2 applied to H .
- 2: Compute E as the $n \times n$ diagonal matrix with elements

$$m^{v-\deg(d_1)}, m^{v-\deg(d_2)}, \dots, m^{v-\deg(d_{\min\{n,q\}})}, 1, \dots, 1$$

in its main diagonal.

- 3: **return** EP .
-

Author Contributions: All authors equally contributed to every aspect of this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by AEI (<https://doi.org/10.13039/501100011033> (accessed on 6 August 2021)), grant number PID2019-110525GB-I00.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Decoding via Encoders

In this appendix we show that the decoding framework for linear codes over Galois rings presented in [6] is still valid for the broader class of chain rings. In addition, we amend a subtle gap in [6].

A splitting structure $(m, \epsilon_0, \dots, \epsilon_{\nu-1})$ is fixed for a chain ring R with maximal ideal m and nilpotency index ν . As we pointed out in Section 3, we may assume $\mathcal{C} = \text{im}(G)$, where

$$G = \begin{pmatrix} G^{(0)} \\ G^{(1)} \\ \vdots \\ G^{(\nu-1)} \end{pmatrix}$$

and, for each $0 \leq i \leq \nu - 1$, $G^{(i)}$ is a $k_i \times n$ matrix whose m -adic decomposition is

$$G^{(i)} = \epsilon_i(\Gamma_i^{(i)})m^i + \epsilon_{i+1}(\Gamma_{i+1}^{(i)})m^{i+1} + \dots + \epsilon_{\nu-1}(\Gamma_{\nu-1}^{(i)})m^{\nu-1},$$

with matrices $\Gamma_j^{(i)}$ whose entries are in F , and $\Gamma_i^{(i)}$ is full rank.

Information is presented as vectors

$$\mathbf{u} = (\mathbf{u}^{(0)} | \dots | \mathbf{u}^{(\nu-1)}),$$

where the m -adic expansion of $\mathbf{u}^{(i)}$ is

$$\mathbf{u}^{(i)} = \epsilon_0(\mathbf{v}_0^{(i)}) + \epsilon_1(\mathbf{v}_1^{(i)})m + \dots + \epsilon_{\nu-1-i}(\mathbf{v}_{\nu-1-i}^{(i)})m^{\nu-1-i}$$

with $\mathbf{v}_j^{(i)} \in F^{k_i}$. Observe that $\mathbf{v}_j^{(i)} = 0$, if $\nu - 1 - i < j$. The reason for this restriction is that terms of higher degree are annihilated in the encoding process. Indeed, let $\mathbf{c} = \mathbf{u}G$. Then

$$\begin{aligned} \mathbf{c} &= \mathbf{u}G \\ &= \sum_{i=0}^{\nu-1} \mathbf{u}^{(i)} G^{(i)} \\ &= \sum_{i=0}^{\nu-1} \sum_{j_1=0}^{\nu-1-i} \epsilon_{j_1}(\mathbf{v}_{j_1}^{(i)})m^{j_1} \sum_{j_2=i}^{\nu-1} \epsilon_{j_2}(\Gamma_{j_2}^{(i)})m^{j_2} \\ &= \sum_{i=0}^{\nu-1} \sum_{j_1=0}^{\nu-1-i} \sum_{j_2=i}^{\nu-1} \epsilon_{j_1}(\mathbf{v}_{j_1}^{(i)})\epsilon_{j_2}(\Gamma_{j_2}^{(i)})m^{j_1+j_2} \\ &= \sum_{i=0}^{\nu-1} \sum_{j_1=0}^{\nu-1-i} \sum_{j=0}^{\nu-1-i} \epsilon_{j_1}(\mathbf{v}_{j_1}^{(i)})\epsilon_{j+i}(\Gamma_{j+i}^{(i)})m^{j_1+j+i} \\ &= \sum_{l=0}^{\nu-1} \sum_{j=0}^l \sum_{i=0}^{l-j} \epsilon_{l-j-i}(\mathbf{v}_{l-j-i}^{(i)})\epsilon_{i+j}(\Gamma_{i+j}^{(i)})m^l \\ &= \sum_{l=0}^{\nu-1} \sum_{j=0}^l (\epsilon_0(\mathbf{v}_0^{(l-j)}) \cdots \epsilon_{l-j-1}(\mathbf{v}_{l-j-1}^{(1)}) \epsilon_{l-j}(\mathbf{v}_{l-j}^{(0)})) \begin{pmatrix} \epsilon_l(\Gamma_l^{(l-j)}) \\ \vdots \\ \epsilon_{1+j}(\Gamma_{1+j}^{(1)}) \\ \epsilon_j(\Gamma_j^{(0)}) \end{pmatrix} m^l \end{aligned} \tag{A1}$$

This is not necessarily the m -adic decomposition of \mathbf{c} , since we do not know if the coefficients of each m^l belong to $\text{im}(\epsilon_l)$. Actually, this is the inaccuracy in Equation (11)

in [6]. However, we may fix this issue, since the knowledge of $\mathbf{v}_j^{(i)}$, for $0 \leq i \leq \nu - 1$ and $0 \leq j \leq \nu - 1 - i$, allows one to recover the codeword.

Let $\mathbf{y} = \mathbf{c} + \mathbf{e}$ with m -adic expansions

$$\begin{aligned} \mathbf{c} &= \sum_{l=0}^{\nu-1} \epsilon_l(\boldsymbol{\zeta}_l)m^l, \\ \mathbf{y} &= \sum_{l=0}^{\nu-1} \epsilon_l(\boldsymbol{\gamma}_l)m^l, \\ \mathbf{e} &= \sum_{l=0}^{\nu-1} \epsilon_l(\boldsymbol{\xi}_l)m^l. \end{aligned}$$

For each $0 \leq l \leq \nu - 1$ it follows that

$$\sum_{i=0}^l \epsilon_i(\boldsymbol{\gamma}_i)m^i - \sum_{i=0}^l \epsilon_i(\boldsymbol{\xi}_i)m^i - \sum_{i=0}^l \epsilon_i(\boldsymbol{\zeta}_i)m^i \in m^{l+1}$$

and, by (A1),

$$\sum_{i=0}^l \epsilon_i(\boldsymbol{\zeta}_i)m^i - \sum_{i=0}^l \sum_{j=0}^i \left(\epsilon_0(\mathbf{v}_0^{(i-j)}) \cdots \epsilon_{i-j}(\mathbf{v}_{i-j}^{(0)}) \right) \begin{pmatrix} \epsilon_i(\Gamma_i^{(i-j)}) \\ \vdots \\ \epsilon_j(\Gamma_j^{(0)}) \end{pmatrix} m^i \in m^{l+1}.$$

Hence, for each $0 \leq l \leq \nu - 1$,

$$\sum_{i=0}^l \epsilon_i(\boldsymbol{\gamma}_i)m^i - \sum_{i=0}^l \epsilon_i(\boldsymbol{\xi}_i)m^i - \sum_{i=0}^l \sum_{j=0}^i \left(\epsilon_0(\mathbf{v}_0^{(i-j)}) \cdots \epsilon_{i-j}(\mathbf{v}_{i-j}^{(0)}) \right) \begin{pmatrix} \epsilon_i(\Gamma_i^{(i-j)}) \\ \vdots \\ \epsilon_j(\Gamma_j^{(0)}) \end{pmatrix} m^i \in m^{l+1} \tag{A2}$$

We use (A2) to describe the decoding framework. We start by computing $\boldsymbol{\xi}_0$ and $\mathbf{v}_0^{(0)}$. Let $C^{(0)} = \text{im}(\Gamma_0^{(0)})$. By (A2) with $l = 0$,

$$\epsilon_0(\boldsymbol{\gamma}_0) - \epsilon_0(\boldsymbol{\xi}_0) - \epsilon_0(\mathbf{v}_0^{(0)})\epsilon_0(\Gamma_0^{(0)}) \in \mathfrak{m}.$$

By (8),

$$\epsilon_0(\boldsymbol{\gamma}_0 - \boldsymbol{\xi}_0 - \mathbf{v}_0^{(0)}\Gamma_0^{(0)}) - \epsilon_0(\boldsymbol{\gamma}_0) + \epsilon_0(\boldsymbol{\xi}_0) + \epsilon_0(\mathbf{v}_0^{(0)}\Gamma_0^{(0)}) \in \mathfrak{m}.$$

These last two equations imply that

$$\epsilon_0(\boldsymbol{\gamma}_0 - \boldsymbol{\xi}_0 - \mathbf{v}_0^{(0)}\Gamma_0^{(0)}) \in \mathfrak{m},$$

i.e.,

$$\boldsymbol{\gamma}_0 - \boldsymbol{\xi}_0 - \mathbf{v}_0^{(0)}\Gamma_0^{(0)} = 0.$$

Observe that $\mathbf{v}_0^{(0)}\Gamma_0^{(0)}$ is a codeword in $C^{(0)}$ and $\boldsymbol{\gamma}_0$ is a received word with error $\boldsymbol{\xi}_0$. If we can decode $\boldsymbol{\gamma}_0$, then we can compute $\boldsymbol{\xi}_0$ and $\mathbf{v}_0^{(0)}$. It follows from (8) that $\boldsymbol{\xi}_0 = \boldsymbol{\gamma}_0 - \boldsymbol{\xi}_0 = \mathbf{v}_0^{(0)}\Gamma_0^{(0)}$.

For the general recursive step, assume that $\boldsymbol{\xi}_i$ for $0 \leq i \leq l - 1$ and $(\mathbf{v}_0^{(j)} \cdots \mathbf{v}_j^{(0)})$ for $0 \leq j \leq l - 1$ are known. Let us describe how to compute $\boldsymbol{\xi}_l$ and $(\mathbf{v}_0^{(l)} \cdots \mathbf{v}_l^{(0)})$. Equation (A2) implies

$$\sum_{i=0}^l \epsilon_i(\boldsymbol{\gamma}_i)m^i - \sum_{i=0}^l \epsilon_i(\boldsymbol{\xi}_i)m^i - \sum_{i=0}^l \sum_{j=0}^i \left(\epsilon_0(\mathbf{v}_0^{(i-j)}) \cdots \epsilon_{i-j}(\mathbf{v}_{i-j}^{(0)}) \right) \begin{pmatrix} \epsilon_i(\Gamma_i^{(i-j)}) \\ \vdots \\ \epsilon_j(\Gamma_j^{(0)}) \end{pmatrix} m^i \in \mathbf{m}^{l+1}.$$

Rearranging its summands we get

$$\begin{aligned} & \sum_{i=0}^{l-1} \epsilon_i(\boldsymbol{\gamma}_i)m^i - \sum_{i=0}^{l-1} \epsilon_i(\boldsymbol{\xi}_i)m^i \\ & - \sum_{i=0}^{l-1} \sum_{j=0}^i \left(\epsilon_0(\mathbf{v}_0^{(i-j)}) \cdots \epsilon_{i-j}(\mathbf{v}_{i-j}^{(0)}) \right) \begin{pmatrix} \epsilon_i(\Gamma_i^{(i-j)}) \\ \vdots \\ \epsilon_j(\Gamma_j^{(0)}) \end{pmatrix} m^i \\ & - \sum_{j=1}^l \left(\epsilon_0(\mathbf{v}_0^{(l-j)}) \cdots \epsilon_{l-j}(\mathbf{v}_{l-j}^{(0)}) \right) \begin{pmatrix} \epsilon_l(\Gamma_l^{(l-j)}) \\ \vdots \\ \epsilon_j(\Gamma_j^{(0)}) \end{pmatrix} m^l + \epsilon_l(\boldsymbol{\gamma}_l)m^l \\ & - \epsilon_l(\boldsymbol{\xi}_l)m^l - \left(\epsilon_0(\mathbf{v}_0^{(l)}) \cdots \epsilon_l(\mathbf{v}_l^{(0)}) \right) \begin{pmatrix} \epsilon_l(\Gamma_l^{(l)}) \\ \vdots \\ \epsilon_0(\Gamma_0^{(0)}) \end{pmatrix} m^l \in \mathbf{m}^{l+1}. \end{aligned} \tag{A3}$$

Equation (A2) also implies

$$\sum_{i=0}^{l-1} \epsilon_i(\boldsymbol{\gamma}_i)m^i - \sum_{i=0}^{l-1} \epsilon_i(\boldsymbol{\xi}_i)m^i - \sum_{i=0}^{l-1} \sum_{j=0}^i \left(\epsilon_0(\mathbf{v}_0^{(i-j)}) \cdots \epsilon_{i-j}(\mathbf{v}_{i-j}^{(0)}) \right) \begin{pmatrix} \epsilon_i(\Gamma_i^{(i-j)}) \\ \vdots \\ \epsilon_j(\Gamma_j^{(0)}) \end{pmatrix} m^i \in \mathbf{m}^l,$$

so, by Proposition 1, there exists $\boldsymbol{\delta}_l \in F^n$ such that

$$\begin{aligned} & \sum_{i=0}^{l-1} \epsilon_i(\boldsymbol{\gamma}_i)m^i - \sum_{i=0}^{l-1} \epsilon_i(\boldsymbol{\xi}_i)m^i \\ & - \sum_{i=0}^{l-1} \sum_{j=0}^i \left(\epsilon_0(\mathbf{v}_0^{(i-j)}) \cdots \epsilon_{i-j}(\mathbf{v}_{i-j}^{(0)}) \right) \begin{pmatrix} \epsilon_i(\Gamma_i^{(i-j)}) \\ \vdots \\ \epsilon_j(\Gamma_j^{(0)}) \end{pmatrix} m^i - \epsilon_l(\boldsymbol{\delta}_l)m^l \in \mathbf{m}^{l+1}. \end{aligned} \tag{A4}$$

Combining (A3) and (A4), we get

$$\begin{aligned} & \left(\epsilon_l(\boldsymbol{\delta}_l) - \sum_{j=1}^l \left(\epsilon_0(\mathbf{v}_0^{(l-j)}) \cdots \epsilon_{l-j}(\mathbf{v}_{l-j}^{(0)}) \right) \begin{pmatrix} \epsilon_l(\Gamma_l^{(l-j)}) \\ \vdots \\ \epsilon_j(\Gamma_j^{(0)}) \end{pmatrix} \right. \\ & \left. + \epsilon_l(\boldsymbol{\gamma}_l) - \epsilon_l(\boldsymbol{\xi}_l) - \left(\epsilon_0(\mathbf{v}_0^{(l)}) \cdots \epsilon_l(\mathbf{v}_l^{(0)}) \right) \begin{pmatrix} \epsilon_l(\Gamma_l^{(l)}) \\ \vdots \\ \epsilon_0(\Gamma_0^{(0)}) \end{pmatrix} \right) m^l \in \mathbf{m}^{l+1}. \end{aligned} \tag{A5}$$

Equation (8) implies

$$\begin{aligned} \epsilon_l \left(\delta_l - \sum_{j=1}^l (v_0^{(l-j)} \dots v_{l-j}^{(0)}) \begin{pmatrix} \Gamma_l^{(l-j)} \\ \vdots \\ \Gamma_j^{(0)} \end{pmatrix} + \gamma_l - \xi_l - (v_0^{(l)} \dots v_l^{(0)}) \begin{pmatrix} \Gamma_l^{(l)} \\ \vdots \\ \Gamma_0^{(0)} \end{pmatrix} \right) \\ - \epsilon_l(\delta_l) + \sum_{j=1}^l (\epsilon_0(v_0^{(l-j)}) \dots \epsilon_{l-j}(v_{l-j}^{(0)})) \begin{pmatrix} \epsilon_l(\Gamma_l^{(l-j)}) \\ \vdots \\ \epsilon_j(\Gamma_j^{(0)}) \end{pmatrix} \\ - \epsilon_l(\gamma_l) + \epsilon_l(\xi_l) + (\epsilon_0(v_0^{(l)}) \dots \epsilon_l(v_l^{(0)})) \begin{pmatrix} \epsilon_l(\Gamma_l^{(l)}) \\ \vdots \\ \epsilon_0(\Gamma_0^{(0)}) \end{pmatrix} \in \mathfrak{m}. \quad (\text{A6}) \end{aligned}$$

Equations (A5) and (A6) imply

$$\epsilon_l \left(\delta_l - \sum_{j=1}^l (v_0^{(l-j)} \dots v_{l-j}^{(0)}) \begin{pmatrix} \Gamma_l^{(l-j)} \\ \vdots \\ \Gamma_j^{(0)} \end{pmatrix} + \gamma_l - \xi_l - (v_0^{(l)} \dots v_l^{(0)}) \begin{pmatrix} \Gamma_l^{(l)} \\ \vdots \\ \Gamma_0^{(0)} \end{pmatrix} \right) m^l \in \mathfrak{m}^{l+1}$$

so, by Proposition 1,

$$\delta_l - \sum_{j=1}^l (v_0^{(l-j)} \dots v_{l-j}^{(0)}) \begin{pmatrix} \Gamma_l^{(l-j)} \\ \vdots \\ \Gamma_j^{(0)} \end{pmatrix} + \gamma_l - \xi_l - (v_0^{(l)} \dots v_l^{(0)}) \begin{pmatrix} \Gamma_l^{(l)} \\ \vdots \\ \Gamma_0^{(0)} \end{pmatrix} = 0$$

which is an equation in F . Let $C^{(l)} = \text{im} \begin{pmatrix} \Gamma_l^{(l)} \\ \vdots \\ \Gamma_0^{(0)} \end{pmatrix}$. We decode the known word $\delta_l + \gamma_l -$

$\sum_{j=1}^l (v_0^{(l-j)} \dots v_{l-j}^{(0)}) \begin{pmatrix} \Gamma_l^{(l-j)} \\ \vdots \\ \Gamma_j^{(0)} \end{pmatrix}$ to compute ξ_l and $(v_0^{(l)} \dots v_l^{(0)})$.

Appendix B. SageMath Code

Here we provide the SageMath [12] code we have implemented to check our algorithms and to produce examples. We have tested our implementation on the Galois rings $R = GR(4, 3), GR(2, 5), GR(3, 3)$ (see Example 1). As far as we have checked, code inside the boxes can be cut and pasted from a PDF file opened with Adobe Acrobat Reader in a Sagemath notebook running under Jupyter. The character `_` represents a white space. Although the code inside the boxes is uglier using this character, it is needed to obtain the correct indentation when pasting the code into Jupyter.

```
p,nu,r_=_2,3,2
#p,nu,r_=_2,5,1
#p,nu,r_=_3,3,1
if_r_=_1:
    F=GF(p)
    R=IntegerModRing(p^nu)
else:
    F.<a>=GF(p^r)
    R.<a>=IntegerModRing(p^nu).extension(F.modulus())
```

The splitting structure is introduced as a list of lists, each one having the images of the elements of the residue field F . The last element is a generator for the maximal ideal.


```

#_p,nu,r_=_2,3,2
splitting_structure_=[ [R(0),R(5*a+4),R(3*a+7),R(4*a+3)],
    [R(0),R(3*a+6),R(1*a+5),R(2*a+7)],
    [R(0),R(5*a+6),R(5*a+1),R(6*a+1)]]+_p]
#_p,nu,r_=_2,5,1
#splitting_structure_=[ [R(0),R(7)],
    [R(0),R(5)],
    [R(0),R(3)],
    [R(0),R(1)],
    [R(0),R(3)]]+_p]
#_p,nu,r_=_3,3,1
#splitting_structure_=[ [R(0),R(7),R(8)],
    [R(0),R(25),R(17)],
    [R(0),R(19),R(11)]]+_p]
mm_=_splitting_structure[nu]

```

The canonical projection $\pi : R \rightarrow F$, splitting maps, m -adic expansion, and its inverse are defined as follows.

```

def_proj(rr):
    if r_==1:
        return(F(rr))
    else:
        return(F(rr.list()))

def_splitting(ff,splitting_list=[R(ele)for ele in F.list()]):
    if ff.parent() != F:
        if (len(splitting_list) !=
            F.cardinality()) or ([proj(r_)
            for r_ in splitting_list] !=
            F.list()):
            return 'Incorrect splitting'
        return splitting_list[F.list().index(ff)]
    elif isinstance(ff,sage.modules.free_module_element.FreeModuleElement):
        ff_ = ff.list()
        aux_ = [splitting(ele,splitting_list) for ele in ff_]
        output_ = vector(R,aux)
        return output
    elif isinstance(ff,sage.matrix.matrix0.Matrix):
        n_rows_ = ff.nrows()
        ff_ = ff.list()
        aux_ = [splitting(ele,splitting_list) for ele in ff_]
        output_ = matrix(R,n_rows,aux)
        return output
    else:
        return 'Type non supported'

def_m_adic(rr,splitting_structure):
    mm_ = splitting_structure[nu]
    if rr.parent() != R:
        rr_ = rr
        output_ = []
        for ii in range(nu):
            if r_==1:
                output_ += [proj(R(ZZ(rr_)//mm^ii))]
            else:
                output_ += [proj(R([ZZ(ele)//mm^ii
                for ele in (rr_.list())])]
                rr_ = rr_ - mm^ii * splitting(output[-1],
                splitting_structure[ii])
            return output
        elif isinstance(rr,
            sage.modules.free_module_element.FreeModuleElement):
            rr_ = rr.list()
            aux_ = [m_adic(ele,splitting_structure) for ele in rr_]
            output_ = [vector(F,[aux[jj][ii]
            for jj in range(len(rr_))])
            for ii in range(nu)]

```

```

#####return_output
#####elif isinstance(rr,
#####sage.matrix.matrix0.Matrix):
#####n_rows=rr.nrows()
#####rr_=rr.list()
#####aux=[m_adic(ele,splitting_structure)for ele in rr_]
#####output=[matrix(F,n_rows,[aux[jj][ii]
#####for jj in range(len(rr_))])
#####for ii in range(nu)]
#####return_output
#####else:
#####return 'Type not supported'

def inv_m_adic(rr,splitting_structure):
#####mm=splitting_structure[nu]
#####return sum(splitting(rr[ii],splitting_structure[ii])*mm^ii
#####for ii in range(nu))

```

In our experiments we have used Goppa codes as efficiently decodable codes over the residue field. The current implementation of Goppa codes in [12] works only for prime fields. Since some of our tests need Goppa codes over \mathbb{F}_4 , we have implemented their construction and decoding by means of the Sugiyama algorithm.

```

from sage.rings.finite_rings.hom_finite_field import FiniteFieldHomomorphism_generic

def GoppaCodeConstructor(n_,t_,F_):
#####m_=ceil(n_.log(F_.cardinality()))
#####k_=n_-2*m_*t_
#####L_=GF(F_.cardinality()^m_)
#####embFL_=FiniteFieldHomomorphism_generic(Hom(F_,L_))
#####secLF_=embFL_.section()
#####V_,_from_V_,_to_V_=L_.vector_space(embFL_,_map=True)
#####R_.<x>=PolynomialRing(L_)
#####tg_=cputime()
#####print('Starting generation')
#####tt_=cputime()
#####g_=R_(x^(2*t_))+_R_.random_element(2*t_-1)
#####while not(g_.is_irreducible()):
#####g_=R_(x^(2*t_))+_R_.random_element(2*t_-1)
#####print('Goppa polynomial',cputime(tt))
######Goppa points
#####tt_=cputime()
#####pts_=[]
#####aux_=L_.list()
#####for ii in range(n_):
#####ind_=ZZ.random_element(len(aux))
#####pts_+=aux[ind]
#####aux.remove(aux[ind])
#####print('Points',cputime(tt))
#####tt_=cputime()
#####Htilde_=matrix.vandermonde(pts_).transpose()[0:2*t_]
#####Htilde_*=diagonal_matrix([g_(ele)^(-1)for ele in pts_])
#####print(cputime(tt))
#####tt_=cputime()
#####aux_=[]
#####for cc in range(Htilde.nrows()):
#####aux2_=Htilde[cc]
#####aux3_=[]
#####for ele in aux2:
#####aux3_+=to_V_(ele).list()
#####aux_+=matrix(aux3).transpose().list()
#####Hhat_=matrix(F_,len(aux)/n_,aux).rref()
#####Paux_=random_matrix(F,n_-k_)
#####while Paux_.is_singular()==True:
#####Paux_=random_matrix(F,n_-k_)
#####H_=Hhat.transpose()*Paux
#####print('Parity check matrix',cputime(tt))
#####print('Generation success',cputime(tg))

```

```

return H_, [L_, g_, pts_, embFL_, secLF_]

def GoppaCodeDecoder(received_, GoppaDecodingData_):
    L_ = GoppaDecodingData_[0]
    g_ = GoppaDecodingData_[1]
    pts_ = GoppaDecodingData_[2]
    t_ = floor(g_.degree()/2)
    R_ = PolynomialRing(L_)
    synd_poly = sum(received_[ii]*R_(x_ - pts_[ii]).inverse_mod(g_))
    for ii in range(len(received_)):
        remainders = [g_, synd_poly]
        coefs = [R_(0), R_(1)]
        while remainders[-1].degree() >= t_:
            cociente, resto = remainders[-2].quo_rem(remainders[-1])
            remainders += [resto]
            coefs += [coefs[-2] - coefs[-1]*cociente]
        locator = coefs[-1]
        evaluator = remainders[-1]
        error_ = []
        for ii in range(len(pts_)):
            root_ = pts_[ii]
            if locator(root_) != 0:
                error_ += [evaluator(root_)/locator.derivative()(root_)]
            else:
                error_ += [L_(0)]
        return vector(error_)

```

The parity check matrix H , as described in (9), is built as follows.

```

length, correction_capability = 60, 3
#length, correction_capability = 256, 7
#length, correction_capability = 20, 2
n, t = length, correction_capability

Decoding_info = []
blocks = []
for ii in range(nu):
    parity_check, decoding_data = GoppaCodeConstructor(n, t, F)
    blocks += [mm^ii*splitting(parity_check,
        splitting_structure[ii])
        + sum(mm^jj*splitting(random_matrix(F,
        parity_check.nrows(),
        parity_check.ncols()),
        splitting_structure[jj])
        for jj in range(ii+1, nu))]
    Decoding_info += [decoding_data]
H = block_matrix(1, blocks)

```

The error vector is built taking random words of bounded Hamming weight at each degree. We assume the codeword is the zero word.

```

error = zero_vector(R, n)
for ll in range(nu):
    xi = zero_vector(F, n)
    while xi.hamming_weight() < t:
        xi = xi.list()
        jj = floor(n*random())
        xi[jj] = F.random_element()
        xi = vector(xi)
        error += splitting(xi,
        splitting_structure[ll])*mm^ll
    received = error
    error, error.hamming_weight()

```

The decoding algorithm (Algorithm 1) is the last piece of code.

```

syndrome = [received * H.subdivision(0, ii) for ii in range(nu)]
sigma = [m_adic(ele, splitting_structure) for ele in syndrome]
Theta = [m_adic(H.subdivision(0, ii),
    splitting_structure) for ii in range(nu)]
xi = []
for ii in range(nu):
    delta = m_adic(
        sum(splitting(sigma[nu-1-ii][jj],
            splitting_structure[jj]) * mm^jj
            for jj in range(nu-1-ii, nu))
        - sum(sum(splitting(xi[l1],
            splitting_structure[l1]) *
            splitting(Theta[nu-1-ii][jj-l1],
            splitting_structure[jj-l1]) * mm^jj
            for ll in range(jj-nu+ii+2))
            for jj in range(nu-1-ii, nu-1))
        - sum(splitting(xi[l1], splitting_structure[l1]) *
            splitting(Theta[nu-1-ii][nu-1-l1],
            splitting_structure[nu-1-l1]) * mm^(nu-1)
            for ll in range(ii), splitting_structure[nu-1])
        rec_aux = Theta[nu-1-ii][nu-1-ii].solve_left(delta)
        error_L = GoppaCodeDecoder([Decoding_info[nu-1-ii][3](ele)
            for ele in rec_aux],
            Decoding_info[nu-1-ii])
        xi += [vector([Decoding_info[nu-1-ii][4](ele)
            for ele in error_L.list()])]
    computed_error = inv_m_adic(xi, splitting_structure)

```

References

1. Hammons, A.R.; Kumar, P.V.; Calderbank, A.R.; Sloane, N.J.; Sole, P. The \mathbb{Z}_4 -linearity of kerdock, preparata, goethals, and related codes. *IEEE Trans. Inf. Theory* **1994**, *40*, 301–319. [\[CrossRef\]](#)
2. Wood, J.A. Duality for modules over finite rings and applications to coding theory. *Am. J. Math.* **1999**, *121*, 555–575. [\[CrossRef\]](#)
3. Gómez-Torrecillas, J.; Hietala-Aho, E.; Lobillo, F.J.; López-Permouth, S.; Navarro, G. Some remarks on non projective Frobenius algebras and linear codes. *Des. Codes Cryptogr.* **2020**, *88*, 1–15. [\[CrossRef\]](#)
4. Feng, C.; Nóbrega, R.W.; Kschischang, F.R.; Silva, D. Communication over finite-ring matrix channels. In Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, 7–12 July 2013; pp. 2890–2894.
5. Feng, C.; Nóbrega, R.W.; Kschischang, F.R.; Silva, D. Communication over finite-chain-ring matrix channels. *IEEE Trans. Inf. Theory* **2014**, *60*, 5899–5917. [\[CrossRef\]](#)
6. Babu, N.S.; Zimmermann, K.-H. Decoding of linear codes over Galois rings. *IEEE Trans. Inf. Theory* **2001**, *47*, 1599–1603. [\[CrossRef\]](#)
7. Greferath, M.; Vellbinger, U. Efficient decoding of \mathbb{Z}_{p^k} -linear codes. *IEEE Trans. Inf. Theory* **1998**, *44*, 1288–1291. [\[CrossRef\]](#)
8. Dinh, H.Q.; Lopez-Permouth, S.R. Cyclic and negacyclic codes over finite chain rings. *IEEE Trans. Inf. Theory* **2004**, *50*, 1728–1744. [\[CrossRef\]](#)
9. McDonald, B.R. *Finite Rings with Identity*; Marcel Dekker: New York, NY, USA, 1974.
10. Albrecht, M.R.; Bernstein, D.J.; Chou, T.; Cid, C.; Gilcher, J.; Lange, T.; Maram, V.; von Maurich, I.; Misoczki, R.; Niederhagen, R.; et al. *Classic McEliece: Conservative Code-Based Cryptography*; Technical Report; NIST's Post-Quantum Cryptography Standardization Project: Gaithersburg, MD, USA, 2020.
11. Brown, W.C. *Matrices over Commutative Rings*; Number 169 in Monographs and Textbooks in Pure and Applied Mathematics; Marcel Dekker, Inc.: New York, NY, USA, 1993.
12. The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.1)*. 2020. Available online: <https://www.sagemath.org> (accessed on 6 August 2021).