Advances in Mathematics of Communications
Volume X, No. 0X, 200X, X–XX
Web site: http://www.aimSciences.org

THE AUTOMORPHISM GROUPS OF LINEAR CODES AND CANONICAL REPRESENTATIVES OF THEIR SEMILINEAR ISOMETRY CLASSES

Thomas Feulner

Department of Mathematics, University of Bayreuth
95440 Bayreuth, Germany

Abstract. The main aim of the *classification* of linear codes is the evaluation of complete lists of representatives of the isometry classes. These classes are mostly defined with respect to *linear* isometry, but it is well known that there is also the more general definition of *semilinear* isometry taking the field automorphisms into account. This notion leads to bigger classes so the data becomes smaller. Hence we describe an algorithm that gives *canonical* representatives of these bigger classes by calculating a unique generator matrix to a given linear code, in a well defined manner.

The algorithm is based on the partitioning and refinement idea which is also used to calculate the canonical labeling of a graph [12] and it similarly returns the automorphism group of the given linear code. The time needed by the implementation of the algorithm is comparable to Leon's program [10] for the calculation of the linear automorphism group of a linear code, but it additionally provides a unique representative and the automorphism group with respect to the more general notion of semilinear equivalence. The program can be used online under `http://www.algorithm.uni-bayreuth.de/en/research/Coding_Theory/CanonicalForm/index.html`.

1. Introduction

A (linear) $[n,k,d]_q$-code is a k-dimensional subspace of $\mathbb{F}_q^n$ such that any two different codewords of C have *Hamming distance* d_{Ham} at least d, i.e. they differ in at least d positions, and there is a pair whose distance is equal to d. The parameter d is called the *minimum distance* of the code C. The *Hamming weight* $wt(c)$ of a codeword $c \in C$ is the number of nonzero entries, i.e. $wt(c) := d_{\mathrm{Ham}}(c,0)$.

A matrix $\Gamma \in \mathbb{F}_q^{k\times n}$ whose rows form a basis of the $[n,k,d]_q$-code C is a *generator matrix* of the code C. The set of all generator matrices is constructed via left multiplication from an arbitrary generator matrix by all invertible $(k\times k)$-matrices $A \in \mathrm{GL}_k(q)$. A mapping $\iota : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is called *isometry*, if it respects the Hamming metric. A mapping $\sigma : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is called *semilinear*, if there exists an automorphism α of $\mathbb{F}_q$ such that, for all $u,v \in \mathbb{F}_q^n, \kappa \in \mathbb{F}_q$ we have $\sigma(u+v) = \sigma(u)+\sigma(v)$ and $\sigma(\kappa u) = \alpha(\kappa)\sigma(u)$.

Two codes C, C' are *equivalent*, if there is a semilinear isometry $\iota : \mathbb{F}_q^n \to \mathbb{F}_q^n$ with $\iota(C) = C'$. Such codes have the same error-correcting capability, and so we only need *representatives of the equivalence classes*. For this purpose we derive *canonical generator matrices* of these representatives. In Section 2 we formulate the

2000 *Mathematics Subject Classification:* Primary: 05E20; Secondary: 20B25, 94B05.

Key words and phrases: automorphism group, canonization, coding theory, error-correcting code, group action, representative, semilinear isometry.

equivalence classes as orbits of a *group action* from the left on the set of generator matrices.

In Section 4 the group action will be *split into two actions*. One action is the permutation of coordinates and therefore an action of the symmetric group S_n. It acts on the orbits (on the set of all generator matrices) of a second group which consists of triples $(A, \varphi; \alpha) \in (\mathrm{GL}_k(q) \times \mathbb{F}_q^{*n}) \rtimes \mathrm{Aut}(\mathbb{F}_q)$[1], where $\mathbb{F}_q^*$ denotes the set of units and $\mathrm{Aut}(\mathbb{F}_q)$ the group of field automorphisms of $\mathbb{F}_q$. We use this decomposition of the group action since only the permutational part really needs an expensive *backtracking procedure* for the canonization. We call the second action the *inner* one, and we will show that the canonization is straightforward. In fact, it is an easy minimization algorithm that amounts to choosing a lexicographically smallest element.

Section 5 describes how to manage the *outer* group action, i.e. the action of the symmetric group S_n. An *automorphism* of a linear code C is a semilinear isometry which maps C onto itself. These mappings form a subgroup $\mathrm{Aut}(C)$ of the group of all semilinear isometries, the stabilizer subgroup of C. The automorphism group is used as a tool for pruning large parts of the search tree. Another very useful tool for pruning is a *homomorphism of group actions*, introduced in Section 3, which corresponds to the *partitioning and refinement* idea [11, 12].

The algorithm that we describe is also based on a backtrack search through the search tree consisting of all elements of the acting group, but in contrast to Leon's (automorphism group) algorithm for linear codes [10] we are just using permutations instead of monomial mappings, although we apply the more general notion of semilinear instead of linear isometry. Furthermore, and even more important, this algorithm can also be used to calculate a unique representative within the semilinear isometry class of a given linear code.

There are some other approaches to solve the code equivalence problem. We mention Sendrier [13] and Bouyukliev [3]. They also do not obtain canonical representatives, but this is urgently needed if we want to built up databases of representatives[2] for each isometry class of $[n, k, d]_q$-codes. Storing canonical representatives in these databases reduces the problem of isometry test to the comparison of two generator matrices instead of testing semilinear isometry for each pair of elements. Using the more general notion of semilinear isometry reduces the size of these databases approximately[3] by the factor $r := |\mathrm{Aut}(\mathbb{F}_q)|$.

2. Code Equivalence

Two linear codes $C_1, C_2 \leq \mathbb{F}_q^n$ are *linearly isometric* if there is a *linear isometry* $\iota : \mathbb{F}_q^n \to \mathbb{F}_q^n$ mapping C_1 onto C_2. This isometry ι can also be expressed as an element $(\varphi; \pi) \in \mathbb{F}_q^{*n} \rtimes S_n$ of the group of *monomial permutations*. The semidirect product $\mathbb{F}_q^{*n} \rtimes S_n$ acts on $\mathbb{F}_q^n$ via

$$(\varphi; \pi)(v_0, \ldots, v_{n-1}) := \left(\varphi_0 v_{\pi^{-1}(0)}, \ldots, \varphi_{n-1} v_{\pi^{-1}(n-1)}\right)$$

and the multiplication within this group is defined by

$$(\varphi; \pi)(\psi; \sigma) := (\varphi\psi_\pi, \pi\sigma) \quad \text{where } (\varphi\psi_\pi)_i := \varphi_i \psi_{\pi^{-1}(i)}, i = 0, \ldots, n-1.$$

[1] a semidirect product of groups; the multiplication is defined in Section 4

[2] This would improve the databases [6] and [15], which give only existence information and information about the construction methods for linear codes of a given parameter set $[n, k, d]_q$.

[3] In [1] numbers of linear and semilinear isometry classes are given.

If $q = p^r$ is not a prime, then the *Frobenius automorphism* $\tau : \mathbb{F}_q \to \mathbb{F}_q$, $x \mapsto x^p$ applied on each coordinate of $\mathbb{F}_q^n$ preserves the Hamming distance, too. Furthermore, linear subspaces are mapped to linear subspaces. Therefore, we include the group of field automorphisms to a more general notion of equivalence of linear codes. We call two codes C_1, C_2 *equivalent* or *semilinearly isometric* if and only if there is a field automorphism $\alpha \in \mathrm{Aut}(\mathbb{F}_q)$ and a linear isometry $\iota : \mathbb{F}_q^n \to \mathbb{F}_q^n$ such that $\iota(\alpha(C_1)) = C_2$. All these mappings again form a group which is isomorphic to $\mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times S_n)$, where the multiplication of elements is defined by

$$(\varphi; (\alpha, \pi))(\psi; (\beta, \sigma)) := (\varphi \cdot \alpha(\psi_\pi); (\alpha\beta, \pi\sigma))$$

It acts in a natural way on $\mathbb{F}_q^n$ and on the set $\mathcal{L}(\mathbb{F}_q^n) := \{C \mid C \leq \mathbb{F}_q^n\}$ of linear subspaces as well.

It is well-known that for $n \geq 3$ the set of all isometries of $\mathbb{F}_q^n$ mapping subspaces to subspaces is the group of semilinear isometries, see [1]. Therefore, the notion of semilinear isometry of linear codes is the most general which can be expressed as a group action on the set of linear subspaces (with respect to the natural action).[4]

Finally, we want to represent the codes by their generator matrices. Let $\mathbb{F}_q^{k\times n,k}$ denote the set of all $(k \times n)$-matrices of rank k, i.e. the set of generator matrices of (linear) $[n, k]_q$-codes.

Since all generator matrices of a given code can be reached by the left multiplication of an arbitrary generator matrix of the code by all invertible matrices $A \in \mathrm{GL}_k(q)$, we add this group in an appropriate way to the one previously constructed. We end up in the group action of $(\mathrm{GL}_k(q) \times \mathbb{F}_q^{*n}) \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times S_n)$ on the set of all $(k \times n)$-matrices $\Gamma \in \mathbb{F}_q^{k\times n,k}$ of rank k. The group action is defined by

$$((A, \varphi); (\alpha, \pi))\Gamma := A\,((\varphi; \pi)\alpha(\Gamma))$$

where α is applied to each entry and $(\varphi; \pi)$ is a linear isometry mapping the rows to a basis of another linear subspace. The multiplication of two group elements is defined by $((A, \varphi); (\alpha, \pi)) \cdot ((B, \psi); (\beta, \sigma)) := ((A\alpha(B), \varphi \cdot \alpha(\psi_\pi)); (\alpha\beta, \pi\sigma))$.

3. Groups and Group Actions

Let G be a group acting on some non-empty set X. The *orbit* of an element $x \in X$ will be denoted by Gx and the set of all orbits by $G\backslash\backslash X := \{Gx \mid x \in X\}$.

Fact 3.1 (Homomorphism Principle, Laue [9])**.** *Let G be a group acting on a set X and H another group acting on Y with surjective mappings $\theta : X \to Y$ and $\varphi : G \to H$. Furthermore let the pair (θ, φ) be a homomorphism of group actions, i.e. $\theta(gx) = \varphi(g)\theta(x),\ \forall\ g \in G, x \in X$. Then*

1. *the stabilizer subgroup G_x of x is a subgroup of $\varphi^{-1}(H_{\theta(x)})$,*
2. *if $T_{H\backslash\backslash Y}$ is a transversal of $H\backslash\backslash Y$, i.e. a minimal, but complete set of orbit representatives, then*

$$T_{G\backslash\backslash X} = \bigcup_{y \in T_{H\backslash\backslash Y}} T_{\varphi^{-1}(H_y)\backslash\backslash\theta^{-1}(y)}$$

is a transversal of $G\backslash\backslash X$.

[4] A non-semilinear isometry must map at least one subspace onto a non-linear block code, violating the definition of a group action. Of course, its still possible that there are semilinearly nonisometric codes and an isometry of $\mathbb{F}_q$ mapping one code onto the other.

We shall mostly apply this theorem with one group G acting on two different sets, where the second set is smaller or easier to handle. A mapping $f : X \to Y$ where (f, id) is a homomorphism of group actions, is called *G-homomorphism*.

The most common way of getting transversal elements of a group action of G on X, is to calculate the smallest elements of the orbits according to some total order on the set X. This would also be possible in our case of a group acting on the set of $(k \times n)$-matrices over $\mathbb{F}_q$. But a naive approach would have to calculate all the matrices within the same orbit in order to decide which one is the minimum.

The Homomorphism Principle provides another transversal $T_{G\backslash\backslash X}$ of the action of G on X. This choice of representatives might be not so intuitive, but the calculation of the unique orbit representative $t \in T_{G\backslash\backslash X}$ in the orbit Gx of an element $x \in X$ becomes much more easier. This choice of representatives corresponds to another total order on the set X

$$x \preceq x' :\Longleftrightarrow \theta(x) < \theta(x') \vee (\theta(x) = \theta(x') \wedge x \leq x'),$$

where we supposed the transversals of minimal elements to be used in both steps of the theorem. The canonization algorithm is as follows:

1. calculate an element $g_0 \in G$ whose image $\varphi(g_0)$ maps $\theta(x)$ to the smallest element $y \in T_{H\backslash\backslash Y}$ in the same orbit,
2. calculate an element $g_1 \in \varphi^{-1}(H_y)$ mapping $g_0 x$ to its minimal representative $g_1 g_0 x$ in the orbit $\left(\varphi^{-1}(H_y)\right) g_0 x$.

Using homomorphisms of group actions reduces the overall complexity about logarithmically. Considering this, we will use the Homomorphism Principle, although it forces us to return transversal elements which are depending on the choice of (θ, φ).

Of course, the calculation of orbit representatives in $H\backslash\backslash Y$ and $\varphi^{-1}(H_y)\backslash\backslash\theta^{-1}(y)$ can be further improved by using this idea iteratively.

Fact 3.2. *Let N be a normal subgroup of a group G, which acts on a set X. Then the factor group $G/N := \{gN \mid g \in G\}$ naturally acts on the set of orbits $N\backslash\backslash X$ and there is a bijection $\Phi(Gx) := (G/N)(Nx)$ between the orbits of G on X and the orbit set $(G/N)\backslash\backslash(N\backslash\backslash X)$.*

The following lemma provides the basis to split the group action of semilinear isometries on generator matrices.

Lemma 3.3. *Let $N \rtimes (G \times H)$ be a semidirect product of groups, where the right factor is a direct product of two groups G, H. We assume that the group $N \rtimes (G \times H)$ is acting on a set X. Then there is a natural bijection between the orbit sets*

$$\Phi : (N \rtimes (G \times H))\backslash\backslash X \to H\backslash\backslash ((N \rtimes G)\backslash\backslash X)$$
$$(N \rtimes (G \times H))\, x \mapsto H((N \rtimes G)\, x).$$

We can calculate a transversal element $t \in (N \rtimes (G \times H))\, x$ in two steps:

1. *Calculate the transversal element $\omega \in H((N \rtimes G)\, x)$ of the outer group action.*
2. *Take the representative $t \in \omega \cap T_{(N \rtimes G)\backslash\backslash X}$ of the inner group action, i.e. $(N \rtimes G)$ acting on X.*

Proof. The group can be expressed equivalently by $(N \rtimes G) \rtimes H$ and the group $(N \rtimes G) \simeq (N \rtimes G) \rtimes \{1_H\}$ on the left is a normal subgroup of the outer semidirect product. Furthermore, the factor group $((N \rtimes G) \rtimes H)\,/(N \rtimes G)$ is isomorphic to H. So Lemma 3.2 provides the bijection.

We use the Homomorphism Principle 3.1 with the projection on H, i.e. $\varphi : N \rtimes (G \times H) \to H$ and the orbit map: $\theta : X \to (N \rtimes G)\backslash\backslash X, x \mapsto (N \rtimes G)\, x$ to show the second part. These maps are surjective and they respect the group actions. For the transversal element ω we conclude that $\varphi^{-1}(H_\omega) \supseteq \varphi^{-1}(\{id_H\}) = (N \rtimes G) \rtimes \{id_H\}$ and $\theta^{-1}(\omega) = \omega$. Therefore the action of $\varphi^{-1}(H_\omega)$ is transitive on $\theta^{-1}(\omega)$ and we just have to pick the orbit representative $t \in \omega$ of the inner group action. □

4. Splitting the group action

Applying Lemma 3.3 to the action of $(\mathrm{GL}_k(q) \times \mathbb{F}_q^{*n}) \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times S_n)$ on $\mathbb{F}_q^{k\times n,k}$ splits the group action into two parts and gives a first algorithmic instruction for the calculation of a canonical element. This will be used in Section 4.2.1 after showing that – in our case – the calculation of transversal elements of the inner group action is easily achieved.

Corollary 1. *There is a bijection between the orbit sets of the group* $\big((\mathrm{GL}_k(q) \times \mathbb{F}_q^{*n}) \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times S_n)\big)$ *acting on the set* $\mathbb{F}_q^{k\times n,k}$ *and the group* S_n *acting on* $\big((\mathrm{GL}_k(q) \times \mathbb{F}_q^{*n}) \rtimes \mathrm{Aut}(\mathbb{F}_q)\backslash\backslash \mathbb{F}_q^{k\times n,k}\big)$.

Since the parameters n, k, q are fixed within the whole article, we suppress them for the sake of clarity in the notion of the inner group according to this break-up and define

$$G^{(sl)} := (\mathrm{GL}_k(q) \times \mathbb{F}_q^{*n}) \rtimes \mathrm{Aut}(\mathbb{F}_q).$$

The multiplication of two elements $((A, \varphi); \alpha), ((B, \psi); \beta)$ is defined by

$$((A, \varphi); \alpha)((B, \psi); \beta) := ((A\alpha(B), \varphi\alpha(\psi)); \alpha\beta).$$

Of course, we can decompose the group of linear isometries analogously using the notation $G^{(l)} := \mathrm{GL}_k(q) \times \mathbb{F}_q^{*n}$ for the normal subgroup.

4.1. The inner group action.

The canonization algorithm for this group action is a generalization of the calculation of the reduced row echelon form of an arbitrary $(k \times n)$-matrix from the left to the right, where we additionally have to take the multiplication of the columns with nonzero field elements and the field automorphisms into account. In fact, we use the Gaussian elemination as a subprocedure. The only difference is that the columns which cannot be mapped to unit vectors will be further minimized.

The group action of $G^{(sl)}$ also induces an action on the matrices with $i \leq n$ columns by taking the first i entries of the vector $\varphi \in \mathbb{F}_q^{*n}$ to define the multiplication on the columns. The *projection*

$$\Pi_i : \mathbb{F}_q^{k\times n} \to \mathbb{F}_q^{k\times i}, (\gamma_0^T, \ldots, \gamma_{n-1}^T) \mapsto (\gamma_0^T, \ldots, \gamma_{i-1}^T).$$

is compatible with these group actions and therefore a $G^{(sl)}$-homomorphism.

We assume that the field $\mathbb{F}_q$ is totally ordered in a canonical way (see Section 6) by some fixed order $\leq$ with $0 < 1 \leq \xi,\ \forall\ \xi \in \mathbb{F}_q^*$. The *lexicographical order* induces a total order on $\mathbb{F}_q^n$. Matrices are interpreted as vectors of columns of length k. The order of these k-dimensional column vectors is the *colexicographical order*, while the order of these n-dimensional vectors of columns is the lexicographic one:

$$(\mathbb{F}_q^{k\times n}, <) := ((\mathbb{F}_q^k, <_{\mathrm{co}})^n, <_{\mathrm{lex}}).$$

The reason for the choice of the colexicographical order is that in this case the unit vectors appear in their natural order. We introduce some notion for those matrices whose projections are minimal in their orbits:

Definition 4.1. Let $1 \leq i \leq n$. A matrix Γ is said to be *i-semicanonical* if

$$\Pi_i(\Gamma) \leq \Pi_i(\widetilde{\Gamma}),\ \forall\, \widetilde{\Gamma} \in G^{(sl)}\Gamma$$

Of course, an i-semicanonical matrix is also i'-semicanonical for $1 \leq i' \leq i$. The matrix Γ^{min} is the unique n-semicanonical element in the orbit $G^{(sl)}\Gamma$, but for an arbitrary i there might be other i-semicanonical elements.

We proceed to calculate an $(i+1)$-semicanonical element inductively starting from an i-semicanonical $\Gamma^{(i)}$. The column with index i is minimized under the stabilizer subgroup $G^{(sl)}_{\Pi_i(\Gamma^{(i)})}$ of the preceding columns. This is once again an application of the Homomorphism Principle 3.1.

The first column of a 1-semicanonical representative $\Gamma^{(1)}$ in the orbit $G^{(sl)}\Gamma$ must be either the zero column or the first unit vector e_0^T since we are allowed to multiply Γ by any invertible matrix. In our notion of the group elements $((A,\varphi);\alpha) \in G^{(sl)}$ we will often suppress components which are the identity elements of their groups.

It is easy to give a generating set for the group $G^{(sl)}_{\Pi_1(\Gamma^{(1)})}$ and we will show that for an arbitrary index i and an arbitrary i-semicanonical representative Γ it remains easy. The generating set can be stored as a list of parameters $(s,\mathfrak{p},t)$ in the following sense:

Definition 4.2. Let $i \in \{1,\ldots,n\}$ and $\Gamma := (\gamma_0^T,\ldots,\gamma_{n-1}^T) \in \mathbb{F}_q^{k\times n}$ be an i-semicanonical representative of its orbit $G^{(sl)}\Gamma$. The projection $\Pi_i(\Gamma)$ uniquely defines

- $s := \mathrm{rk}(\Pi_i(\Gamma)) = \dim(\mathrm{span}(\gamma_0,\ldots,\gamma_{i-1}))$,
- $t := \min\{t' \in \mathbb{N} \mid \mathrm{Aut}(\mathbb{F}_q)_{\Pi_i(\Gamma)} = \langle \tau^t\rangle\}$ and
- $\mathfrak{p} := \{p_0,\ldots,p_{l-1}\}$ a partition of the set $\{0,\ldots,s-1\}$ such that for all $j \in \{0,\ldots,i-1\}$ there exists exactly one index $m \in \{0,\ldots,l-1\}$ such that $\mathrm{supp}(\gamma_j) \subseteq p_m$, and $\mathfrak{p}$ is the smallest[5] such partition.

Lemma 4.3. *Let $\Gamma \in \mathbb{F}_q^{k\times n}$ be an i-semicanonical matrix and define $(s,\mathfrak{p},t)$ in the sense of Definition 4.2. The stabilizer $G^{(sl)}_{\Pi_i(\Gamma)}$ of $\Pi_i(\Gamma)$ is generated by*

1. *the map $\tau^t \in \mathrm{Aut}(\mathbb{F}_q)$,*
2. *all matrices $\left(\begin{smallmatrix} I_s & A_1 \\ 0 & A_2 \end{smallmatrix}\right) \in \mathrm{GL}_k(q)$ where I_s denotes the $s\times s$ identity matrix, $A_1 \in \mathbb{F}_q^{s\times(k-s)}$ and $A_2 \in \mathrm{GL}_{k-s}(q)$,*
3. *arbitrary column multiplications of zero columns of Γ,*
4. *all column multiplication vectors $\varphi \in \mathbb{F}_q^{*n}$, with $\varphi_j = 1,\ \forall\ 0 \leq j < i$, and*
5. *the* crossed row-column-multiplications *for all $p_j \in \mathfrak{p}, \mu \in \mathbb{F}_q^*$:*
   $RC(p_j,\mu,\Gamma) := (D,\varphi) \in \mathrm{GL}_k(q) \times \mathbb{F}_q^{*n}$

$$D := \mathrm{diag}(d_0,\ldots,d_{k-1}) \text{ with } d_l = \begin{cases} \mu, & l \in p_j \\ 1, & \text{else} \end{cases}, l \in \{0,\ldots,k-1\}$$

$$\varphi_l := \begin{cases} \mu^{-1}, & \mathrm{supp}(\gamma_l) \subseteq p_j \\ 1, & \text{else} \end{cases}, l \in \{0,\ldots,n-1\}$$

[5] according to the partial order:
$\{p_0,\ldots,p_{l-1}\} \preceq \{q_0,\ldots,q_{l'-1}\} :\Longleftrightarrow \forall\ m \in \{0,\ldots,l-1\}\ \exists\ m' \in \{0,\ldots,l'-1\} : p_m \subseteq q_{m'}$

Algorithm 1 SEMICANONICAL – Matrix Minimizer for $i \in \{0, \ldots, n\}$

Input: $\Gamma = (\gamma_0^T, \ldots, \gamma_{n-1}^T) \in \mathbb{F}_q^{k \times n}$

Output: $\Gamma^{(i)} = ({\gamma_0^{(i)}}^T, \ldots, {\gamma_{n-1}^{(i)}}^T) \in G^{(sl)}\Gamma$ with $\Gamma^{(i)}$ i-semicanonical

Output: $(s, \mathfrak{p}, t)$ defining $G^{(sl)}_{\Pi_i(\Gamma^{(i)})}$ according to Lemma 4.3

procedure SEMICANONICAL(i, Γ)
  if $i = 0$ **then**
    return $(\Gamma, (0, \{\emptyset\}, 1))$;// the group $G^{(sl)}$
  end if
  $(\Gamma^{(i)}, (s, \mathfrak{p}, t)) \leftarrow$ SEMICANONICAL$(i-1, \Gamma)$;// minimize the first $i-1$ columns
  if $\gamma_{i-1}^{(i)} \notin \operatorname{span}(\gamma_0^{(i)}, \ldots, \gamma_{i-2}^{(i)})$ **then** // do Gaussian elimination
    Calculate $B \in G(i-1, \Gamma^{(i)})$ with $B\left(\gamma_{i-1}^{(i)}\right)^T = e_s^T$;
    $\Gamma^{(i)} \leftarrow B\Gamma^{(i)}$;
    $(s, \mathfrak{p}) \leftarrow (s+1, \mathfrak{p} \cup \{\{s\}\})$;
  else
    union_index $\leftarrow -1$;
    for $j \leftarrow s-1$ **to** 0 **do**
      if $\Gamma_{j,i-1}^{(i)} = 0$ **then**
        continue;// find first nonzero entry in this column
      end if
      Determine λ with $j \in p_\lambda$;
      if union_index $= -1$ **then** // first nonzero entry
        union_index $\leftarrow \lambda$;
        $\gamma_{i-1}^{(i)} \leftarrow {\Gamma_{j,i-1}^{(i)}}^{-1} \cdot \gamma_{i-1}^{(i)}$; // normalize the column
      else if $\lambda \neq$ union_index **then** // the first element in $p_\lambda \cap \operatorname{supp}(\gamma_{i-1}^{(i)})$
        $\Gamma^{(i)} \leftarrow RC(p_\lambda, (\Gamma_{j,i-1}^{(i)})^{-1}, \Gamma^{(i)}) \cdot \Gamma^{(i)}$;// map this entry to 1
        $p_{\text{union\_index}} \leftarrow p_{\text{union\_index}} \cup p_\lambda$;// join the subsets of the partition
        $p_\lambda \leftarrow \emptyset$;
      else // minimize with the help of the remaining field automorphisms
        Choose $m \in \{1, \ldots, \frac{r}{t}\} : \tau^{tm}(\Gamma_{j,i-1}^{(i)}) \leq \tau^{tx}(\Gamma_{j,i-1}^{(i)}),\ \forall\ x \in \{1, \ldots, \frac{r}{t}\}$;
        $\Gamma^{(i)} \leftarrow \tau^{tm}(\Gamma^{(i)})$;// minimize this entry
        $t \leftarrow t \cdot \min\{t' > 0 \mid \tau^{tt'}(\Gamma_{j,i-1}^{(i)}) = \Gamma_{j,i-1}^{(i)}\}$; // fix the minimized entry
      end if
    end for
    $\mathfrak{p} \leftarrow \{p_l \in \mathfrak{p} \mid p_l \neq \emptyset\}$;
  end if
  return $\left[\Gamma^{(i)}, (s, \mathfrak{p}, t)\right]$;
end procedure

Proof. The straightforward proof is left to the reader. □

Algorithm 1 calculates for each $i \in \{0, \ldots, n\}$ an i-semicanonical generator matrix $\Gamma^{(i)}$. The case $i = 0$ is added for technical reasons. Furthermore it returns the parameter set $(s^{(i)}, \mathfrak{p}^{(i)}, t^{(i)})$ which uniquely defines the stabilizer $G^{(sl)}_{\Pi_i(\Gamma^{(i)})}$.

By this recursion starting with SEMICANONICAL(n, Γ), we calculate the lexicographically smallest representative $\Gamma^{(n)}$ in the orbit $G^{(sl)}\Gamma$ for a given $\Gamma \in \mathbb{F}_q^{k\times n}$.

Proof of Algorithm 1. We show by induction that the algorithm is correct. For $i = 0$ the algorithm of course returns the right answer. The induction further shows that the matrix $\Gamma^{(i)}$ returned by SEMICANONICAL$(i-1, \Gamma)$ contains the unit vectors $e_0^T, \ldots, e_{s-1}^T$, $s := \mathrm{rk}(\Pi_{i-1}(\Gamma^{(i)}))$ and these columns of course remain unchanged by the stabilizer.

We have to distinguish two different cases. In the first case, the vector $\gamma_{i-1}^{(i)}$ is not an element of $\mathrm{span}(\gamma_0^{(i)}, \ldots, \gamma_{i-2}^{(i)})$ of the preceding columns. The stabilizer returned by SEMICANONICAL$(i-1, \Gamma)$ is equal to $G^{(sl)}_{\Pi_{i-1}(\Gamma^{(i)})}$ containing the invertible matrices $\left(\begin{smallmatrix} I_s & A_1 \\ 0 & A_2 \end{smallmatrix}\right)$. Therefore, we can obtain the vector e_s^T by the left multiplication with an appropriate matrix $B \in G^{(sl)}_{\Pi_{i-1}(\Gamma^{(i)})}$ and $\Pi_i(B\Gamma^{(i)})$ must be minimal in its orbit. Of course, s must be increased by one and we have to append the subset $\{s\}$ to the partition $\mathfrak{p}$.
In the second case, the column with index $i-1$ is linearly dependent on the preceding ones. This implies that its entries with index $\geq s$ must be equal to zero. We search for the last nonzero entry. If it is found, the column is treated by the block starting at line 17 which transform this entry to 1 by the multiplication of the column by the inverse field element.
Now, each set p_l of the partiton $\mathfrak{p}$ which is not equal to $p_{\text{union\_index}}$ and has a nontrivial intersection with the support $\mathrm{supp}(\gamma_{i-1}^{(i)})$ gives us the opportunity to transform the entry $\Gamma^{(i)}_{j,i-1}$ with $j = \max(p_l \cap \mathrm{supp}(\gamma_{i-1}^{(i)}))$ to 1. This is implemented by the crossed row-column-multiplication in line 21.
Finally, the remaining subgroup of the field automorphisms allows us to minimize all the remaining entries in $\mathrm{supp}(\gamma_{i-1}^{(i)})$. This step is described in line 25. It is easy to see that the operations on the parameters $(s, \mathfrak{p}, t)$ ensures the conditions of Definition 4.2. □

With some minor modifications Algorithm 1 can be used to minimize the columns in an arbitrary fixed order and to return the applied group element. If we are interested in a minimal orbit representative of the group $G^{(l)}$, we just have to remove the lines 24–28. The parameter t of the stabilizer subgroup becomes redundant in this case.

Example 1. Let $\mathbb{F}_4 = \{0, 1, x, x^2\}$ with $x^2 + x + 1 = 0$ and $0 < 1 < x < x^2$. We want to calculate SEMICANONICAL$(n, \Gamma^{(0)})$ with $\Gamma^{(0)} := \begin{pmatrix} 1 & x & x^2 & 1 & x & 1 \\ x & 0 & x^2 & 1 & 1 & 1 \\ x & 1 & 0 & 1 & 0 & x \end{pmatrix}$. We call Algorithm 1 recursively until reaching level $i = 1$. See also Table 1 for the output in each step.

For $i = 1, 2$ we have to follow the instructions given by the block starting at line 7, mapping the first two columns of Γ to the unit vectors e_0^T and e_1^T.

The next recursive call of the algorithm for $i = 3$ has to use the line 10 and the following. Starting with $j = 1$, the if-condition in line 17 is satisfied. We multiply the column with $\Gamma^{(3)\,-1}_{1,2} = (x^2)^{-1} = x$ and set union_index = 1. For $j = 0$ we have to perform a crossed row-column-multiplication $RC\left(p_0, (x^2)^{-1}, \Gamma^{(3)}\right)$ following the lines starting at 21. Since both nonzero entries of the column $(\gamma_2^{(3)})^T$ are now equal to 1 they will not be changed by the Frobenius automorphism τ.

i	performed operation in step i	$\Gamma^{(i)}$	$s^{(i)}, \mathfrak{p}^{(i)}, t^{(i)}$
1	$\begin{pmatrix} 1 & 0 & 0 \\ x & 1 & 0 \\ x & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & x & x^2 & 1 & x & 1 \\ 0 & x^2 & x & x^2 & x & x^2 \\ 0 & x & 1 & x^2 & x^2 & 0 \end{pmatrix}$	1 $\{\{0\}\}$ 1
2	$\begin{pmatrix} 1 & x^2 & 0 \\ 0 & x & 0 \\ 0 & x^2 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & x & x^2 & x^2 & x^2 \\ 0 & 1 & x^2 & 1 & x^2 & 1 \\ 0 & 0 & 0 & 1 & x & x \end{pmatrix}$	2 $\{\{0\},\{1\}\}$ 1
3	$\left(\begin{pmatrix} x & & \\ & 1 & \\ & & 1 \end{pmatrix}, (x^2, 1, x, 1, 1, 1)\right)$	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & x^2 & 1 \\ 0 & 0 & 0 & 1 & x & x \end{pmatrix}$	2 $\{\{0,1\}\}$ 1
4	$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 1 & 0 & x^2 & x^2 \\ 0 & 1 & 1 & 0 & 1 & x^2 \\ 0 & 0 & 0 & 1 & x & x \end{pmatrix}$	3 $\{\{0,1\},\{2\}\}$ 1
5	$\left(\begin{pmatrix} x^2 & & \\ & x^2 & \\ & & 1 \end{pmatrix}, (x, x, x, 1, x, 1); \tau\right)$	$\begin{pmatrix} 1 & 0 & 1 & 0 & x & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & x^2 \end{pmatrix}$	3 $\{\{0,1,2\}\}$ 2
6	$(1, 1, 1, 1, 1, x)$	$\begin{pmatrix} 1 & 0 & 1 & 0 & x & x \\ 0 & 1 & 1 & 0 & 1 & x \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$	3 $\{\{0,1,2\}\}$ 2

Table 1. Output in Example 1

In step $i = 4$ we perform a Gaussian elimination. On level $i = 5$ we first multiply the column $(\gamma_4^{(5)})^T$ by x^2 resulting in $(x, x^2, 1)^T$. The entry x^2 is mapped to 1 by the crossed row-column-multiplication $RC\left(p_0, (x^2)^{-1}, \Gamma^{(5)}\right)$. Now the last two columns are $\begin{pmatrix} x^2 & 1 \\ 1 & 1 \\ 1 & x \end{pmatrix}$ and the entry x^2 can be mapped by the Frobenius automorphism τ to x. For $i = 6$, we just multiply the last column by x.

4.2. The outer group action. For the outer group action – the action of the symmetric group S_n on a set X – there is a well-known approach for canonization applying partitions and refinements (Leon [11] , McKay [12]). We will follow this idea.

At first glance, the complexity of the set X in our case might lead to more difficulties. But as we have seen above, we are able to manage this set of orbits by n-semicanonical representatives efficiently. Thereby we avoid to take the bigger group of monomial permutations interpreted as a subgroup of $S_{n(q-1)}$ to formulate the backtrack search algorithm [10]. This reduces the size of the search tree and allows us to use the more appropriate concept of semilinear isometry for the notion of equivalence of linear codes.

4.2.1. *A first naive Algorithm.* With the help of Algorithm 1 we are already able to calculate the automorphism group of a linear code C given by a generator matrix Γ and a uniquely determined generator matrix Γ^{can} of all semilinearly isometric codes to C. We just have to apply all permutations $\pi \in S_n$ on the generator matrix Γ and have to call Semicanonical$(n, \pi\Gamma)$ returning a uniquely determined matrix

$\Gamma^{(n,\pi)}$. We define the canonical element according to Lemma 3.3

$$\Gamma^{can} := \min\{\Gamma^{(n,\pi)} \mid \pi \in S_n\} = \min\left(\left((\mathrm{GL}_k(q) \times \mathbb{F}_q^{*n}) \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times S_n)\right)\Gamma\right).$$

Two different permutations $\pi, \sigma \in S_n$ with minimal n-semicanonical representatives $\Gamma^{can} = \Gamma^{(n,\pi)} = ((A,\varphi);(\alpha,\pi))\Gamma$ and $\Gamma^{can} = \Gamma^{(n,\sigma)} = ((B,\psi);(\beta,\sigma))\Gamma$ give rise to an automorphism

$$((A,\varphi);(\alpha,\pi))^{-1} \cdot ((B,\psi);(\beta,\sigma))$$

of Γ. The automorphism group $\mathrm{Aut}(\Gamma)$ is generated by all those products together with the stabilizer of the inner group action

$$G_\Gamma^{(sl)} = ((A,\varphi);(\alpha,\pi))^{-1} G_{\Gamma^{can}}^{(sl)} ((A,\varphi);(\alpha,\pi)),$$

which is also returned by Algorithm 1.

4.2.2. *Backtrack Search.* Of course, the naive approach described above is only applicable for very small parameters n, because the cardinality of the symmetric group S_n grows exponentially. The next step is to merge permutations which imply similar i-semicanonical representatives.

Let G be a subgroup of S_n. We define the *pointwise stabilizer* of the first i points by $G^{(i)} = G_{(0,\ldots,i-1)} = G_0 \cap \ldots \cap G_{i-1}$.

We use the following example to motivate an improved approach, which will allow us to handle codes with higher length efficiently.

Example 2. Let $\Gamma \in \mathbb{F}_q^{3\times n}$ with $n \geq 4$, where the first four columns are given by

$$\Pi_4(\Gamma) = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

For any permutation $\pi \in S_n$ which permutes those four columns and maps the last column $(0,0,1)^T$ to a position further left the matrix $\pi\Gamma$ will contain three linearly independent columns on its first positions. For this reason the n-semicanonical representative $\Gamma^{(n,\pi)}$ will contain the 3×3 identity matrix on its first three columns. Due to the lexicographic order on the columns and

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

we conclude that the n-semicanonical representatives $\Gamma^{(n,\pi)}$ cannot be equal to Γ^{can} and we are able to detect this difference by comparing the projections on the first three columns of any of the 3-semicanonical matrices $\Gamma^{(3,\pi)} \in G^{(sl)}(\pi\Gamma)$:

$$\Pi_3(\Gamma^{can}) \leq \Pi_3(\Gamma^{(3,id)}) \leq \Pi_3(\Gamma) < \Pi_3(\Gamma^{(3,\pi)}).$$

Let $\sigma \in S_n^{(3)}$ be another permutation. We have $\Pi_3(\sigma\Gamma') = \Pi_3(\Gamma')$ for any $\Gamma' \in \mathbb{F}_q^{k\times n}$ which implies $\sigma\Gamma^{(3,\pi)}$ to be a 3-semicanonical representative in $G^{(sl)}(\sigma\pi\Gamma)$. This information can be used to conclude that $\sigma\pi$ does not lead to the canonical representative, since $\Pi_3(\Gamma^{(3,\sigma\pi)}) = \Pi_3(\sigma\Gamma^{(3,\pi)}) = \Pi_3(\Gamma^{(3,\pi)}) > \Pi_3(\Gamma^{can})$.

As soon as we have found a permutation $\pi \in S_n$ which leads to an i-semicanonical representative $\Gamma^{(i,\pi)}$ whose projection $\Pi_i(\Gamma^{(i,\pi)})$ is greater than the projection of our candidate for the canonical representative Γ^{can} we can eliminate the whole right coset $S_n^{(i)}\pi$ from our search. In the case $\Pi_i(\Gamma^{(i,\pi)}) < \Pi_i(\Gamma^{can})$ the candidate Γ^{can} cannot be the canonical representative and we replace Γ^{can} by $\Gamma^{(n,\pi)}$.

We substitute the naive brute force approach by a systematical enumeration of all permutations. The coset $S_n^{(i)}\pi$ consists of all permutations whose preimages of the first i positions are given by the preimages of π:

$$S_n^{(i)}\pi = \{\sigma \in S_n \mid \sigma^{-1}(j) = \pi^{-1}(j),\ \forall\ 0 \leq j < i\}.$$

Prescribing additionally the preimage of the point i gives us $n - i$ possibilities to extend the vector $(\pi^{-1}(0), \ldots, \pi^{-1}(i-1))$ by a further preimage. This corresponds to the disjoint decomposition of the coset

$$S_n^{(i)}\pi = \bigcup_{\sigma \in T^{(i)}} S_n^{(i+1)}\sigma\pi$$

where $T^{(i)}$ is a right transversal of $S_n^{(i+1)}$ in $S_n^{(i)}$. We interprete the cosets $S_n^{(i)}\pi$ as nodes of a rooted tree structure with root node $S_n^{(0)}id = S_n$. For $i < n$, each node $S_n^{(i)}\pi$ is parent of the nodes $S_n^{(i+1)}\sigma\pi, \sigma \in T^{(i)}$. We label the arc $(S_n^{(i)}\pi, S_n^{(i+1)}\sigma\pi)$ by the transversal element σ. The leafs of this tree are the cosets $S_n^{(n)}\pi = \{\pi\}$. Using a depth-first-search procedure to visit all leafs of this search tree corresponds to the enumeration of all permutations, such that the elements of an arbitrary coset $S_n^{(i)}\pi$ are consecutive. As we have seen in the example above, we can skip the permutations $S_n^{(i)}\pi$ in certain situations, i.e. we can *prune* the subtree rooted in $S_n^{(i)}\pi$.

Algorithm 2 describes the backtrack search algorithm to calculate the canonical generator matrix of an equivalence class of a linear code C given by its generator matrix Γ. We replace the nodes $S_n^{(i)}\pi$ by i-semicanonical representatives of the orbits $G^{(sl)}(\pi\Gamma)$, remembering that the first i columns stay fixed and the remaining columns may be further permuted. The root node of this search tree is therefore equal to Γ. The sons of this node – calculated in Algorithm 3 – represent all possibilities to move an arbitrary column of the matrix Γ to the first position. Increasing i by 1 in the next step, see line 13 of Algorithm 3, tells us to fix this column in the whole subtree rooted in this node.

By calling Algorithm 1, see line 5, we receive i-semicanonical representatives in each step. We always keep a candidate Γ^{can} for the canonical representative, which is n-semicanonical and therefore also i-semicanonical for all $i \leq n$. If the projections $\Pi_i(\Gamma^{can})$ and $\Pi_i(\Gamma^{(i,\pi)})$ do not coincide, then all orbits $G^{(sl)}(\sigma\pi\Gamma), \sigma \in S_n^{(i+1)}$ are different from the orbit $G^{(sl)}\Gamma^{can}$ – see Example 2. We have to distinguish two

Algorithm 2 CodeCan – Linear code canonization algorithm

Input: $\Gamma \in \mathbb{F}_q^{k\times n}$
Output: $\pi^{can} \in S_n$ Transporter Element
Output: $\Gamma^{can} \in \mathbb{F}_q^{k\times n}$ canonical representative of the semilinear isometry class
Output: $A = \mathrm{Aut}(C)$ the automorphism group of C

1: **procedure** CodeCan(Γ)
2: $A \leftarrow \{id\}$;
3: $\Gamma^{can} \leftarrow$ NIL;
4: CodeCanStep$(1, id, \Gamma)$;// construct the sons of the root node
5: **return** $(\pi^{can}, \Gamma^{can}, A)$;
6: **end procedure**

Algorithm 3 CODECANSTEP – One step of the canonization algorithm

Input: Global Variables (defined in Algorithm 2) $\Gamma^{can},\ \pi^{can},\ A$
Input: $i \in \{1,\dots,n\},\ \pi \in S_n,\ \Gamma \in \mathbb{F}_q^{k\times n}$

```
1: procedure CodeCanStep(i, π, Γ)
2:   for j ← i − 1 to n − 1 do
3:     Γ′ ← (i − 1, j)Γ; // swap columns i − 1 and j
4:     σ ← (i − 1, j);
5:     (Γ′, __) ← Semicanonical(i, Γ′); // take an i-semicanonical representative
       of the orbit
6:     if Γ^can ≠ NIL ∧ Π_i(Γ′) > Π_i(Γ^can) then
7:       continue;// prune this subtree, it has no canonical elements
8:     end if
9:     if Γ^can ≠ NIL ∧ Π_i(Γ′) < Π_i(Γ^can) then // the candidate is not canonical
10:      Γ^can ← NIL; // will be newly set by the next leaf, see line 16
11:    end if
12:    if i < n then
13:      CodeCanStep(i + 1, σπ, Γ′); // next level, depth first search
14:    else // a leaf node
15:      if Γ^can = NIL then // a new candidate for the canonical element
16:        Γ^can ← Γ′;
17:        π^can ← σπ;
18:      else // an automorphism
19:        A ← ⟨A ∪ {π^{-1}σ^{-1}π^can}⟩;
20:      end if
21:    end if
22:  end for
23: end procedure
```

different cases. The first is that the projection of this node is smaller than the projection of Γ^{can} – see line 9. In this case, we replace the candidate Γ^{can} by $\Gamma^{(i,\pi)}$ (more precisely: by the n-semicanonical orbit representative which is calculated in the next leaf). If the projection of this node is greater than the projection of the candidate, we can skip this subtree, since no representative would be found there, see line 6.

The calculation of unique representatives of the linear isometry classes and the linear automorphism group can be done with the same backtracking, where we just have to replace the calculation of the i-semicanonical representatives, see line 5 of Algorithm 3, by the linear version of Algorithm 1.

It is not necessary to store the elements of $G^{(sl)}$ applied to Γ during the backtracking. The calculation is done after finishing Algorithm 2 for the generators of A and the element π^{can} using an adapted version of Algorithm 1.

5. Pruning the Search Tree

In this section we describe the possibilities to further reduce the size of the search tree and we introduce the two most important functions in the final implementation of Algorithm 3.

5.1. Pruning subtrees by the Homomorphism Principle. In order to use the Homomorphism Principle, we define suitable G-homomorphisms

$$f : G^{(sl)} \backslash\backslash \mathbb{F}_q^{k\times n} \to X$$

for appropriate subgroups $G \leq S_n$ and codomains X which will be used in Algorithm 3. All homomorphisms which will be used are of some special type:

Definition 5.1. Let $G \leq S_n$. We call a G-homomorphism $f : G^{(sl)} \backslash\backslash \mathbb{F}_q^{k\times n} \to Y^n$ *G-signature* if it assigns to each column index $j \in \{0, \ldots, n-1\}$ a value $y_j \in Y$ and the operation of G on the codomain is just the natural action on n-vectors $(y_0, \ldots, y_{n-1}) \in Y^n$.

The next definition introduces a generalization of the pointwise stabilizer. Afterwards we will show that only these special subgroups of S_n will occur as stabilizers.

Definition 5.2. A strictly increasing sequence $0 = \alpha_0 < \alpha_1 < \ldots < \alpha_m = n$ of natural numbers gives an *(ordered) partition* of the set $\{0, \ldots, n-1\}$

$$[\alpha_0, \ldots, \alpha_m] := \{\{\alpha_0, \ldots, \alpha_1 - 1\}, \ldots, \{\alpha_{m-1}, \ldots, \alpha_m - 1\}\}$$

where we call the subsets $\{\alpha_i, \ldots, \alpha_{i+1} - 1\}$ the *blocks* or *cells* of the partition. The *canonical Young subgroup* of the partition $[\alpha_0, \ldots, \alpha_m]$ is defined to be the intersection of the setwise stabilizer subgroups of its cells

$$S_{[\alpha_0,\ldots,\alpha_m]} := \bigcap_{i=1}^{m} (S_n)_{\{\alpha_{i-1},\ldots,\alpha_i-1\}}.$$

For two partitions $\alpha = [\alpha_0, \ldots, \alpha_m]$ and $\beta = [\beta_0, \ldots, \beta_{m'}]$ of $\{0, \ldots, n-1\}$ we say α is *finer* than β – or a *refinement* of β – if $S_{[\alpha_0,\ldots,\alpha_m]} \leq S_{[\beta_0,\ldots,\beta_{m'}]}$.[6]

The intersection of two canonical Young subgroups of S_n is again a canonical Young subgroup. The same holds for the stabilizer $S_n^{(i)} = S_{[0,1,\ldots,i-1,i,n]}$ of the first i points. Suppose we reached some node on level i in our backtracking procedure and the stabilizer of the predecessor node on level $i-1$ is some canonical Young subgroup. On level i the column $i-1$ is additionally fixed and we have to build the intersection of this group with $S_n^{(i)}$ resulting in a canonical Young subgroup $S_{[\alpha_0,\ldots,\alpha_m]}$.

Next, we use the homomorphism principle by applying an $S_{[\alpha_0,\ldots,\alpha_m]}$-signature with codomain Y^n, which is ordered lexicographically or colexicographically according to some total order on the set Y. The canonical elements should be the smallest elements in the orbits. Calculating the orbit representatives $y^{\text{can}} \in S_{[\alpha_0,\ldots,\alpha_m]} f(G^{(sl)}\Gamma)$ of the image $f(G^{(sl)}\Gamma)$ is easily achieved by sorting the entries of the vector $f(G^{(sl)}\Gamma)$ within the blocks $\{\alpha_{i-1} \ldots, \alpha_i - 1\}, i = 1, \ldots, m'$ given by $[\alpha_0, \ldots, \alpha_m]$. Now we are only allowed to permute equal entries of the vector y^{can} within the same blocks. But these entries are consecutive and therefore the stabilizer $\left(S_{[\alpha_0,\ldots,\alpha_m]}\right)_{y^{\text{can}}}$ is a canonical Young subgroup as well.

Induction shows that the occurring subgroups of S_n are always canonical Young subgroups. This is just the group theoretic validity of McKay's algorithm [12].

Algorithm 4 describes some additional code lines which can be placed after line 11 of Algorithm 3 to apply the Homomorphism Principle. The sequence $(f_0^{(i)}, \ldots, f_{v-1}^{(i)})$ of homomorphisms has to be chosen in some unique way. The signatures which are explicitly used are stated next.

[6] or equivalently: each block $\{\alpha_i, \ldots, \alpha_{i+1} - 1\}, 0 \leq i \leq m-1$ is contained in some block $\{\beta_j, \ldots, \beta_{j+1} - 1\}, 0 \leq j \leq m' - 1$

Algorithm 4 REFINE – The Homomorphism Principle

1: **for** $l \leftarrow 0$ **to** $v-1$ **do**
2: $y \leftarrow f_l^{(i)}(G^{(sl)}\Gamma')$;
3: Calculate $\rho \in S_{[\alpha_0,\dots,\alpha_m]} : \rho y = y^{\text{can}} \in S_{[\alpha_0,\dots,\alpha_m]}(y)$ canonical representative;
4: **if** $\Gamma^{can} \neq NIL \wedge y^{\text{can}} < f_l^{(i)}(G^{(sl)}\Gamma^{can})$ **then**
5: $\Gamma^{can} \leftarrow NIL$; // a new candidate for the canonical element
6: **end if**
7: **if** $\Gamma^{can} \neq NIL \wedge y^{\text{can}} > f_l^{(i)}(G^{(sl)}\Gamma^{can})$ **then**
8: **break** j; // Prune the whole subtree below this node
9: **end if**
10: $\Gamma' \leftarrow \rho\Gamma'$;
11: $\sigma \leftarrow \rho\sigma$;
12: Calculate partition $[\beta_0,\dots,\beta_{m'}] : S_{[\beta_0,\dots,\beta_{m'}]} = \left(S_{[\alpha_0,\dots,\alpha_m]}\right)_{y^{\text{can}}}$;
13: $[\alpha_0,\dots,\alpha_m] \leftarrow [\beta_0,\dots,\beta_{m'}]$;
14: **end for**

We further have to modify line 2 of Algorithm 3: Suppose $S_{[\alpha_0,\dots,\alpha_m]}$ is the remaining, operating group on level $i-1$, i.e. $[\alpha_0,\dots,\alpha_m] = [1,\dots,i-1,\alpha_i,\dots,\alpha_m]$. This information is provided via an additional input parameter $[\alpha_0,\dots,\alpha_m]$ of the function CODECANSTEP. The upper bound of the for loop can be replaced by $\alpha_i - 1$ since only the column indices in the cell $\{\alpha_{i-1},\dots,\alpha_i-1\}$ need to be mapped to position $i-1$. The operating group is then replaced by $S_{[\alpha_0,\dots,\alpha_m]} \cap S_n^{(i)}$ in order to fix position $i-1$ in this subtree.

Lemma 5.3. *With the help of i-semicanonical representatives $\Gamma := (\gamma_0^T,\dots,\gamma_{n-1}^T)$ of each orbit, we define the $S_n^{(i)}$-signature $f_0^{(i)} : G^{(sl)} \backslash\backslash \mathbb{F}_q^{k\times n,k} \to (\{0,1\}^k \cup \{\infty\})^n$,*

$$\left(f_0^{(i)}(G^{(sl)}\Gamma)\right)_j := \begin{cases} \infty, & \text{if } \gamma_j \notin \text{span}(\gamma_0,\dots,\gamma_{i-1}) \\ \delta_{\text{supp}(\gamma_j)}, & \text{else} \end{cases}$$

where $(\delta_{\text{supp}(\gamma_j)})_m := \begin{cases} 1, & \text{if } m \in \text{supp}(\gamma_j) \\ 0, & \text{else} \end{cases}$ *is the binary representation of* $\text{supp}(\gamma_j)$.

Proof. The map is well defined since any other i-semicanonical representative is reached by the multiplication of an element in $G^{(sl)}_{\Pi_i(\Gamma)}$. These elements respect the support of a column $\gamma_j \in \text{span}(\gamma_0,\dots,\gamma_{i-1})$. Of course, $f_0^{(i)}$ is an $S_n^{(i)}$-signature. □

Example 3. Let $i=3$ and $\mathbb{F}_q = \mathbb{F}_4 = \{0,1,x,x^2\}$. The matrix

$$\Gamma := \begin{pmatrix} 1 & 0 & 0 & x & x^2 & x^2 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & x & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

is 3-semicanonical. Taking this matrix as input to $f_0^{(3)}$ we get:

$$f_0^{(3)}\left(G^{(sl)}\Gamma\right) = \left(\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}1\\1\\1\\0\end{pmatrix}, \infty, \begin{pmatrix}1\\0\\1\\0\end{pmatrix}\right)$$

Canonizing the image tells us to apply the permutation $(3,4,5)$ in order to sort the columns $3,4$ and 5 of the image. This gives us a discrete partition of the column indices, i.e. a trivial stabilizer subgroup $S_{[0,1,2,3,4,5,6]} = \{id\}$. We conclude that there is only one leaf $G^{(sl)}((3,4,5)\Gamma)$ in the pruned subtree below the original node instead of $6 = |S_6^{(3)}|$ nodes in the original search tree.

The next signature is based on the fact that the entries of two i-semicanonical representatives $\Gamma' = ((A,\varphi);\alpha)\Gamma$ with $((A,\varphi);\alpha) \in G^{(sl)}_{\Pi_i(\Gamma)}$ cannot be arbitrary. Let $(s,\mathfrak{p},t)$ be the parameter set of Definition 4.2 defined by $\Pi_i(\Gamma)$. Now, take a column γ_l^T of Γ which lies in the span of the first i columns. Suppose $\Gamma_{\kappa,l} \neq 0$ for some $\kappa < s$ and κ^* is defined by $\kappa \in p_{\kappa^*} \in \mathfrak{p}$. Applying (A,φ) to $\alpha(\Gamma)$ multiplies all rows $m \in p_{\kappa^*}$ by the same nonzero field element[7] and therefore the quotients satisfy $\Gamma'_{\kappa,l} \cdot {\Gamma'_{m,l}}^{-1} = \alpha\left(\Gamma_{\kappa,l} \cdot \Gamma_{m,l}^{-1}\right)$, where α is an element of $\langle \tau^t \rangle \leq \mathrm{Aut}(\mathbb{F}_q)$.

Lemma 5.4. *Let $0 \leq \kappa \leq k-1$ be arbitrary and $\infty \notin \mathbb{F}_q$ one more symbol. The mapping*

$$\left(f^{(i)}_{1,\kappa}(G^{(sl)}\Gamma)\right)_l := \begin{cases} \langle \tau^t \rangle \left(\Gamma_{\kappa,l} \cdot \Gamma_{m,l}^{-1}\right)_{m \in p_{\kappa^*}}, & \gamma_l^T \in \mathrm{span}(\gamma_0,\ldots,\gamma_{i-1}) \wedge \Gamma_{\kappa,l} \neq 0 \\ \infty, & \text{else} \end{cases}$$

defined on i-semicanonical representatives $\Gamma := (\gamma_0^T,\ldots,\gamma_{n-1}^T)$ is well-defined and an $S_n^{(i)}$-signature.

Example 4. We start with the 4-semicanonical representative

$$\Gamma = \begin{pmatrix} 1 & 0 & 1 & 0 & x^2 & 1 & 0 \\ 0 & 1 & 1 & 0 & x & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & x^2 & 1 \end{pmatrix} \in \mathbb{F}_4^{3\times 7}$$

The stabilizer of $\Pi_4(\Gamma)$ is defined by the parameters $(3,\{\{0,1\},\{2\}\},1)$. We calculate the image in the case $\kappa = 0$. The orbits $\langle \tau^t \rangle \left(\Gamma_{\kappa,l} \cdot \Gamma_{m,l}^{-1}\right)_{m \in p_{\kappa^*}}$ will be represented by the lexicographically smallest element.

$$f^{(4)}_{1,0}\left(G^{(sl)}\Gamma\right) = \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \infty, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \infty, \begin{pmatrix} 1 \\ x \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \infty\right)$$

For the column $j = 4$ we have to take the minimization by the field automorphism into account since the quotient vector originally is equal to $(1,\ x^2)^T$.

A wide range of $S_n^{(i)}$- and $S_{[\alpha_0,\ldots,\alpha_{m'}]}$-homomorphisms can be constructed by taking *code invariants* $\mathcal{V} : \mathcal{L}(\mathbb{F}_q^n) \to X$, i.e. mappings which return the same result for semilinearly isometric codes.

The most important example in our case might be the *weight enumerator* $W_C(x) := \sum_{c \in C} x^{wt(c)}$ of the code C. By puncturing and shortening the code – which are defined below – in selected subsets of the column indices, we are able to construct a lot of different homomorphisms.

Definition 5.5. Let Γ be a generator matrix of a linear code C of length n. For a subset $J \subset \{0,\ldots,n-1\}$

[7] Because the matrix components of the generators of $G^{(sl)}_{\Pi_i(\Gamma)}$ multiply these entries by the same nonzero element and an addition of nonzero elements cannot happen.

- the *punctured* code C_J is constructed by replacing the entries at positions $j \in J$ of all codewords by zero, and
- the *shortened* code $C_{\setminus J}$ of the code C is the subset of all codewords $(c_0, \ldots, c_{n-1}) \in C$ with $c_j = 0 \ \forall \ \ j \in J$.

In contrast to the standard definition of shortening and puncturing a code, we do not remove the positions indexed by J and therefore C_J as well as $C_{\setminus J}$ are linear codes of length n. This avoids problems in defining the signatures in the next lemma:

Lemma 5.6. *Let J_1, J_2 be disjoint subsets of $\{0, \ldots, i-1\}$, $C(\Gamma)$ the code generated by Γ and $\mathcal{V} : \mathcal{L}(\mathbb{F}_q^n) \to X$ a code invariant, then the mapping*

$$f^{(i)}_{2,J_1,\setminus J_2} : G^{(sl)} \backslash\backslash \mathbb{F}_q^{k\times n,k} \to X^n, \ G^{(sl)}\Gamma \mapsto (x_0^\Gamma, \ldots, x_{n-1}^\Gamma)$$
$$x_j^\Gamma := \mathcal{V}\left(\left((C(\Gamma)_{J_1})_{\setminus J_2}\right)_{\{j\}}\right), j \in \{0, \ldots, n-1\}$$

is an $S_n^{(i)}$-signature. The same is true for $f^{(i)}_{3,J_1,\setminus J_2}$ defined analogously by shortening at the coordinates $j \in \{0, \ldots, n-1\}$.

Proof. The mappings are independent of the choice of the representative Γ since we are using an invariant in the definition of x_j^Γ. Let $\sigma \in S_n^{(i)}$ be arbitrary. It is obviously true that $\sigma(C_J) = \sigma(C)_{\sigma(J)}$ and $\sigma(C_{\setminus J}) = \sigma(C)_{\setminus\sigma(J)}$ for arbitrary subsets $J \subset \{0, \ldots, n-1\}$ and arbitrary linear codes C of length n. This and the fact $\sigma(J_i) = J_i, i = 1, 2$ are used to prove the homomorphism property:

$$\begin{aligned} x_j^{\sigma\Gamma} =& \mathcal{V}\left(\left((C(\sigma\Gamma)_{J_1})_{\setminus J_2}\right)_{\{j\}}\right) = \mathcal{V}\left(\left(((\sigma C(\Gamma))_{J_1})_{\setminus J_2}\right)_{\{j\}}\right) \\ =& \mathcal{V}\left(\left(((\sigma C(\Gamma))_{\sigma(J_1)})_{\setminus\sigma(J_2)}\right)_{\sigma\{\sigma^{-1}(j)\}}\right) = \mathcal{V}\left(\sigma\left(((C(\Gamma))_{J_1})_{\setminus J_2}\right)_{\{\sigma^{-1}(j)\}}\right) \\ =& \mathcal{V}\left(\left(((C(\Gamma))_{J_1})_{\setminus J_2}\right)_{\{\sigma^{-1}(j)\}}\right) = x_{\sigma^{-1}(j)}^\Gamma \end{aligned}$$

$$\implies f^{(i)}_{2,J_1,\setminus J_2}(G^{(sl)}(\sigma\Gamma)) = \sigma f^{(i)}_{2,J_1,\setminus J_2}(G^{(sl)}\Gamma). \qquad \square$$

Example 5. Let $\Gamma = \begin{pmatrix} 1&0&0&0&0&1&1 \\ 0&1&0&0&1&1&0 \\ 0&0&1&0&1&0&1 \\ 0&0&0&1&1&1&1 \end{pmatrix}$ be the generator matrix of the $[7,4,3]_2$–Hamming Code where the first two columns are already fixed. We calculate the image of $f^{(2)}_{2,\emptyset,\setminus\{0,1\}}$ taking the weight enumerator as invariant of the code.

words of the shortened code $C_{\setminus\{0,1\}}$		j	$\left(f^{(2)}_{2,\emptyset,\setminus\{0,1\}}\right)_j$
		0	$1+2x^3+x^4$
$(\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ )$		1	$1+2x^3+x^4$
$(\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ )$		2	$1+2x^2+x^4$
$(\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ )$	$\rightsquigarrow$	3	$1+x^2+2x^3$
$(\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ )$		4	$1+x^2+2x^3$
		5	$1+x^2+2x^3$
		6	$1+x^2+2x^3$

In order to reduce the computational complexity of the signature using the weight enumerator as invariant, we modify it to only count the number of codewords equal to the minimum distance of the code. This set of words can be calculated once

for the root node and adapted to the actually considered permutation to get an $S_n^{(i)}$-signature on any level i.

Since our acting group is already reduced to a subgroup $S_{[\alpha_0,\ldots,\alpha_m]} \leq S_n^{(i)}$, we are not restricted to code invariants at all. We can also construct $S_{[\alpha_0,\ldots,\alpha_m]}$-signatures using $S_{[\alpha_0,\ldots,\alpha_m]}$-invariants of the code. The most important example of such an invariant is the *block weight enumerator* of the partition $[\alpha_0,\ldots,\alpha_m]$:

Definition 5.7. Let $[\alpha_0,\ldots,\alpha_m]$ be a partition of n. We define the *block weight enumerator* of this partition to be the polynomial

$$W_C^{[\alpha_0,\ldots,\alpha_m]}(x_0,\ldots,x_{m-1}) := \sum_{c\in C} \prod_{i=0}^{m-1} x_i^{wt_{[\alpha_i,\alpha_{i+1}]}(c)}$$

where $wt_{[\alpha_i,\alpha_{i+1}]}(c) := |\{j \mid \alpha_i \leq j < \alpha_{i+1} \wedge c_j \neq 0\}|$ is the Hamming weight of the codeword $c \in C$ in the block $\{\alpha_i,\ldots,\alpha_{i+1}-1\}$.

Another $S_n^{(i)}$-signature may be formulated by Leon's idea [10] of taking invariant subsets of the code C. The adaption of this idea is possible, but not applied. We also did not investigate invariants using the hull of a linear code, see [13].

5.2. Pruning subtrees by automorphisms. We introduce complete labeled branchings due to Jerrum [8] to manage the group of known automorphisms within our backtrack search algorithm. This gives us a test for pruning isomorphic subtrees. The type of a permutation $g \in G \leq S_n, g \neq \mathrm{id}$ is defined as the pair $\mathrm{type}(g) := (i,j)$ where $j := g(i)$ is the image of the first nonfixed position $i \in \{0,\ldots,n-1\}$.

A tuple (B,γ) is called *labeled branching* of G, if

1. B is a branching on the set of nodes $\{0,\ldots,n-1\}$, i.e. an acyclic, directed graph with at most one arc ending at each node. Each arc (i,j) fulfills the condition $i < j$.
2. $\gamma = (\gamma_0,\ldots,\gamma_{n-1}) \in (S_n)^n$ is a vector of permutations with
   - if (i,j) is an arc in B, then $\sigma_{ij} := \gamma_j^{-1}\gamma_i$ is of $\mathrm{type}(\sigma_{ij}) = (i,j)$ and
   - the set $\{\sigma_{ij} \mid (i,j) \text{ is an arc in } B\}$ generates G.

A branching B defines a partial order on the set $\{0,\ldots,n-1\}$ via:

$$i \preceq j :\Longleftrightarrow \text{there is a directed path from } i \text{ to } j \text{ in } B.$$

If each set $T^{(i)} = \{\sigma_{ij} \mid i \preceq j\}$ is a complete set of left coset representatives of $G^{(i)}/G^{(i+1)}$, we say the labeled branching is *complete*.

Fact 5.8 (Jerrum [8])**.** *Let A be a subgroup of S_n and (B,γ) a complete labeled branching of A. A permutation $\pi \in S_n$ is lexicographically minimal in πA, if and only if π is a topological sorting of B, i.e. $i \preceq j \Longrightarrow \pi(i) \leq \pi(j)$.*

We will store the group $A \leq S_n$ of known automorphisms of $G^{(sl)}\Gamma$ using a complete labeled branching. During our backtrack search algorithm we are only interested in visiting the transversal of lexicographically minimal elements t of each coset of A, since all other elements in tA lead to isomorphic copies of $G^{(sl)}(t\Gamma)$.

Suppose we have reached level i in our backtrack search algorithm. Let π be the permutation which we have already applied to the root node and $S_{[\alpha_0,\ldots,\alpha_m]} \leq S_n^{(i)}$ the remaining permutations. Lemma 5.9 gives us a test, if there are topological sortings in the subtree $S_{[\alpha_0,\ldots,\alpha_m]}\pi$. The lemma tightens a test of Gugisch [7] formulated for $S_n^{(i)}\pi$.

Lemma 5.9. *Let (B, γ) be a complete labeled branching of a subgroup A of S_n, $[\alpha_0, \ldots, \alpha_m]$ a partition of n and $\pi \in S_n$ arbitrary. There are topological sortings of B in the coset $S_{[\alpha_0,\ldots,\alpha_m]}\pi$ if and only if for any arc (i, j) in B the image $\pi(i)$ is not allowed to lie in a cell further right than the cell containing $\pi(j)$, i.e.*

$$\big[\pi(i) \in \{\alpha_v, \ldots, \alpha_{v+1} - 1\} \wedge \pi(j) \in \{\alpha_u, \ldots, \alpha_{u+1} - 1\}\big] \Longrightarrow v \leq u.$$

Proof. We suppose π is not fulfilling the condition above. Then, there is an arc (i, j) and for the corresponding indices $u, v \in \{0, \ldots, m-1\}$ the index v is greater than u. For all $\sigma \in S_{[\alpha_0,\ldots,\alpha_m]}$ also the images $\sigma\pi(j) \in \{\alpha_u, \ldots, \alpha_{u+1} - 1\}$ and $\sigma\pi(i) \in \{\alpha_v, \ldots, \alpha_{v+1} - 1\}$ must lie in the same sets. Now

$$\sigma\pi(j) \leq \alpha_{u+1} - 1 < \alpha_v \leq \sigma\pi(i)$$

and no topological sorting of B can be found in $S_{[\alpha_0,\ldots,\alpha_m]}\pi$.

For the converse, we construct a permutation $\sigma \in S_{[\alpha_0,\ldots,\alpha_m]}$ such that in each block $\{\alpha_w, \ldots, \alpha_{w+1} - 1\}$, $w \in \{0, \ldots, m-1\}$ of the partition the sequences of preimages $\big(\pi^{-1}\sigma^{-1}(\alpha_w), \ldots, \pi^{-1}\sigma^{-1}(\alpha_{w+1} - 1)\big)$ are ascending.
The permutation $\sigma\pi$ is a topological sorting of B: Let (i, j) be an arbitrary arc in B and define u, v like above:

- if $v < u$, then the images $\sigma\pi(j)$ and $\sigma\pi(i)$ also lie in the sets $\{\alpha_u, \ldots, \alpha_{u+1}-1\}$ and $\{\alpha_v, \ldots, \alpha_{v+1} - 1\}$, respectively. Thus, $\sigma\pi(i) < \sigma\pi(j)$.
- if $v = u$, then the way we have chosen σ gives us $\pi^{-1}\sigma^{-1}(x) = i < j = \pi^{-1}\sigma^{-1}(y)$ for some $x, y \in \{\alpha_u, \ldots, \alpha_{u+1} - 1\} : x < y$, which implies $\sigma\pi(i) = x < y = \sigma\pi(j)$.

□

Once we found a new automorphism in our search, we add it to the group A and update the complete labeled branching to this bigger group. This reduces the set of transversal elements of S_n/A and thereby makes the test more restrictive. In fact, the effort is always reduced by the order $|A|$ of the group of known automorphisms. The test is applied in each iteration of Algorithm 4 just before calling the homomorphism and once again, when returning to Algorithm 3.

6. Numbering the Field Elements

Within the article the field was just specified to be totally ordered in some way with the zero element to be smallest and $1 \leq \mu$, $\forall\ \mu \in \mathbb{F}_q^*$. Of course, different ways of numbering the field elements lead to different lexicographical orders on the generator matrices and therefore they result in different i-semicanonical orbit representatives calculated by the algorithm. To realize and utilize databases of canonical forms traceable and independent from the different isomorphic field representations, we need a canonical order of the field, too.

Of course, prime fields $\mathbb{F}_p$, can be ordered by identifying the residue classes $x+(p)$ by their representatives $x \in \{0, \ldots, p-1\}$ and the natural order $0 < 1 < \ldots < p-1$. If $q = p^r$ with $r > 1$, we choose the *Conway polynomial* [14] f of degree r over $\mathbb{F}_p$ to define the field extension, since these polynomials are used in most computer algebra systems by default. Let α be a root of this polynomial. Then any element $\mu \in \mathbb{F}_q$ can be uniquely expressed by

$$\mu = \sum_{i=0}^{r-1} a_i \alpha^i \quad \text{with } a_i \in \mathbb{F}_p, i = 0, \ldots, r-1.$$

This representation of the field elements and the above identification of residue classes and their representatives in $\{0,\ldots,p-1\}$ give us a bijection

$$N^\alpha : \mathbb{F}_q \to \{0,\ldots,q-1\}, \quad \sum_{i=0}^{r-1} a_i\alpha^i \mapsto \sum_{i=0}^{r-1} a_i p^i$$

and therefore a total order on $\mathbb{F}_q$: For all $\lambda,\mu \in \mathbb{F}_q : \lambda <_\alpha \mu :\Longleftrightarrow N^\alpha(\lambda) < N^\alpha(\mu)$.

We have to guarantee that the result of our canonization algorithm is independent of the choice of the root α.

Lemma 6.1. *Let α,β be two roots of a monic irreducible polynomial f over $\mathbb{F}_p$ of degree r and $q := p^r$. Then*

$$(N^\alpha)^{-1} \circ N^\beta \in \mathrm{Aut}(\mathbb{F}_q).$$

Proof. For an arbitrary field element $\mu = \sum_{i=0}^{r-1} b_i\beta^i \in \mathbb{F}_q$ we have

$$\left((N^\alpha)^{-1} \circ N^\beta\right)(\mu) \quad = (N^\alpha)^{-1}\left(\textstyle\sum_{i=0}^{r-1} b_i p^i\right) = \sum_{i=0}^{r-1} b_i\alpha^i,$$

defining an automorphism of $\mathbb{F}_q$. □

The algorithm works with a labeling of the field elements according to the root α. If another root β of the same irreducible polynomial f is used to construct the input, the canonical representative will be the same. This is guaranteed by the lemma above. Taking a root from another monic, irreducible polynomial of degree r may lead to a different canonical generator matrix.

7. Conclusion

We have described an algorithm which is able to calculate a uniquely determined generator matrix in the set of all generator matrices of semilinearly isometric codes to a given linear code. Further, this algorithm can also be used to calculate the automorphism group of this code in the more general notion of semilinear isometry.

We point out that the algorithm should be applied to a parity check matrix of the code in the case $k > \frac{n}{2}$, since the effort is mainly depending on the parameter k. The result can be adapted to the generator matrix of the code easily.

A first implementation of the algorithm is very promising, although it is not containing a heuristic selection of $S_n^{(i)}$-homomorphisms which should be applied on level i. Furthermore, it is easy to expand the set of homomorphisms and there are still a lot which have not been tested at all. It is planned to use the algorithm for building up a database of canonical generator matrices. But since the canonical form is depending on the choice of homomorphisms, this work is not yet done.

We used the algorithm to calculate the automorphism group of the *CCZ-Code* [4] $C_f^{(n)}$ for the almost perfect nonlinear (APN-) function $f^{(n)} : \mathbb{F}_2^n \to \mathbb{F}_2^n, x \mapsto x^3$ with parameters $[2^n, 2n+1, d^{(n)}]_2$. Here are the times in seconds needed on a single core of a 2.4 GHz Intel Quad 2 processor. Times are compared with the implementation of Leon's algorithm in the computer algebra system Magma [2]:

n	d	$\lvert\mathrm{Aut}(C)\rvert$	Leon's algorithm	the new algorithm
9	240	2354688	28	45
10	480	10475520	20	6
11	992	46114816	exceeding Magma's	14400
12	1984	201277440	memory limit	360
14	8064	3757867008	of 2.3 GB	2330000

For $n = 12$ the number of words of minimal weight is 1397760 and 350MB of memory are sufficient during the calculation process[8]. For $n = 14$ there are even 22368256 words with Hamming weight 8064 and the program uses less than 4GB of memory. We are still looking for some appropriate signatures for a better performance in the odd case.

Moreover, we applied the algorithm to the database provided by Kohnert [5] and compared runtimes for the calculation with Leon's algorithm. In most cases time spent in the two different algorithms only differed by a factor less than 2 independent from the parameter set $[n, k, d]_q$, but there are counter examples in both directions. Moreover, this comparison discounts the fact that our algorithm solves a more general problem.

The adaption of the algorithm to linear codes over finite chain rings R is possible, but the performance will mainly depend on the minimization algorithm of the inner group action. In the special case of $R = \mathbb{Z}_4$ we are already able to give such an algorithm, which is hardly more difficult than the procedure for finite fields.

REFERENCES

[1] A. Betten, M. Braun, H.Fripertinger, A. Kerber, A. Kohnert and A. Wassermann, "Error-Correcting Linear Codes. Classification By Isometry And Applications," Springer, Berlin, 2006.

[2] W. Bosma, J. Cannon and C. Playoust, *The Magma algebra system. I: The user language*, J. Symbolic Comput., **24** (1997), 235–265.

[3] I. Bouyukliev, *About the code equivalence*, Ser. Coding Theory Cryptol., **3** (2007), 126–151.

[4] C. Bracken, E. Byrne, N. Markin and G. McGuire, *New families of quadratic almost perfect nonlinear trinomials and multinomials*, Finite Fields Appl., **14** (2008), 703–714.

[5] M. Braun, A. Kohnert and A. Wassermann, *Optimal linear codes from matrix groups*, IEEE Trans. Inform. Theory, **51** (2005), 4247–4251.

[6] M. Grassl, "Bounds on the minimum distance of linear codes and quantum codes," Online available at `http://www.codetables.de`.

[7] R. Gugisch, "Konstruktion von Isomorphieklassen orientierter Matroide," Ph.D Thesis, University of Bayreuth, 2005.

[8] M. Jerrum, *A compact representation for permutation groups*, J. Algorithms, **7** (1986), 60–78.

[9] R. Laue, *Constructing objects up to isomorphism, simple 9-designs with small parameters*, in "Algebraic combinatorics and applications" (eds. A. Betten et al.), Springer, (2001), 232–260.

[10] J. S. Leon, *Computing automorphism groups of error-correcting codes*, IEEE Trans. Inform. Theory, **28** (1982), 496–511.

[11] J. S. Leon, *Partitions, refinements, and permutation group computation*, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., **28** (1997), 123–158.

[12] B. D. McKay, *Practical graph isomorphism*, Congr. Numer., **30** (1981), 45–87.

[13] N. Sendrier, *Finding the permutation between equivalent linear codes: The support splitting algorithm*, IEEE Trans. Inform. Theory, **46** (2000), 1193–1203.

[14] A. Scheerhorn, *Trace- and norm-compatible extensions of finite fields*, Appl. Algebra Engrg. Comm. Comput., **3** (1992), 199–209.

[15] W. Schmid and R. Schürer *MinT: a database for optimal net parameters*, in "Monte Carlo and Quasi-Monte Carlo Methods 2004" (eds. H. Niederreiter and D. Talay), Springer, (2006), 457–469.

Received XXX 200X; revised XXX 200X.

E-mail address: thomas.feulner@uni-bayreuth.de

[8]Because of Magma's memory limit we used an exhaustive search to get all the vectors of minimum weight. In the general implementation, we use a call of Magma routines instead. The codeword c is stored via the information vector $v \in \mathbb{F}_2^k : v\Gamma = c$.